# BAB I

# WIDGET-WIDGET DASAR PADA FLUTTER

Pembelajaran yang akan dilakukan pada bab ini adalah pembelajaran mendasar mengenai *framework* Flutter yang merupakan salah satu jenis *framework* yang digunakan untuk mengembangkan suatu aplikasi perangkat *mobile.* Selain pemberian materi, praktikan juga akan melakukan praktek secara langsung untuk mengimplementasikan materi yang telah disampaikan*.*

## Tujuan Praktikum

|  |  |
| --- | --- |
| **Tujuan** | **Penjelasan** |
| **Mengetahui dan memahami widget-widget dasar yang ada pada Flutter** | Pada bab ini, praktikan diharapkan dapat mengetahui dan memahami widget-widget dasar dalam membangun aplikasi *mobile* dengan Flutter |
| **Mengetahui cara membuat *project* baru pada Flutter dan memahami struktur *project* Flutter** | Praktikan diharapkan dapat mengetahui dan memahami cara membuat *project* Flutter untuk pertama kali dan mengetahui struktur *folder* yang ada pada Flutter |

## Persyaratan Praktikum

Disarankan praktikan menggunakan *hardware* dan *software* sesuai pada dokumentasi ini. Apabila terdapat versi yang lumayan lampau dari versi yang direkomendasikan atau *hardware* yang lawas maka sebaiknya bertanya kepada Asisten mengajar *shift*.

|  |  |
| --- | --- |
| ***HARDWARE* YANG DIBUTUHKAN PRAKTIKUM** | **JENIS** |
| **PC / Laptop CPU** | **≥ 4 *Cores*** |
| **PC / Laptop RAM** | **≥ 8 GB** |
| **PC / Laptop *Storage*** | **≥ 10 GB** |

|  |
| --- |
| ***SOFTWARE* YANG DIBUTUHKAN PRAKTIKUM** |
| **Android *Studio* / *Visual Studio Code*** |

## Materi Praktikum

Pada praktikum MCS bab 1, praktikan akan diajarkan mengenai widget-widget dasar yang ada pada *flutter* yang digunakan untuk membentuk suatu halaman aplikasi. Selain itu, praktikan juga akan langsung mengimplementasikan widget-widgettersebut dalam membentuk sebuah aplikasi. Pada pemrograman flutter, terdapat beberapa widgetdasar yang harus dipahami terlebih dahulu sebelum programmer memulai pengembangan sebuah aplikasi menggunaakan flutter. Berikut beberapa widgetdasar yang harus dikuasai:

### MaterialApp()

MaterialApp() merupakan widgetpaling dasar yang berfungsi sebagai fondasi yang berisikan konfigurasi umum dari aplikasi yang akan dibangun. Untuk menggunakan widgetini, pengguna harus melakukan *import* terhadap **'package:flutter/material.dart'.** Berikut merupakan struktur dari widgetMaterialApp():

|  |
| --- |
| MaterialApp(  title: ...,  debugShowCheckedModeBanner: ...,  theme: ...,  home: ....  routes: {...}  //dan lain lain  ); |

### Scaffold()

Scaffold() merupakan widget umum yang digunakan untuk membentuk halaman pada sebuah aplikasi. Widgetini memiliki berbagai macam properti, seperti **appBar:, body:, drawer:, floatingActionButton:, bottomNavigationBar:** dan lain-lain. Berikut merupakan struktur dari widgetscaffold():

|  |
| --- |
| Scaffold(  appBar: ...,  body: ...,  floatingActionButton: ...,  drawer: ...,  bottomNavigationBar: ...,  ) |

Untuk menggunakan berbagai jenis widgetyang tersedia pada flutter, programmer perlu melakukan *extends* terhadap *class* yang dibentuk dengan menggunakan **StatelessWidget()** atau **StatefulWidget().** Kedua widgettersebut memiliki perbedaan dan kegunaannya masing-masing. Berikut beberapa perbedaan antara **StatelessWidget()** dengan **StatefulWidget()**:

**StatelessWidget():**

1. Hanya fokus pada tampilan
2. Tidak dapat menangani perubahan tampilan. Misal terdapat angka 1 pada halaman kemudian ingin dilakukakn perubahan tertentu dengan aktivitas tertentu pada angka tersebut, dengan StatelessWdget hal tersebut tidak bisa dilakukan
3. Dibuat dengan extends dari StatelessWidget

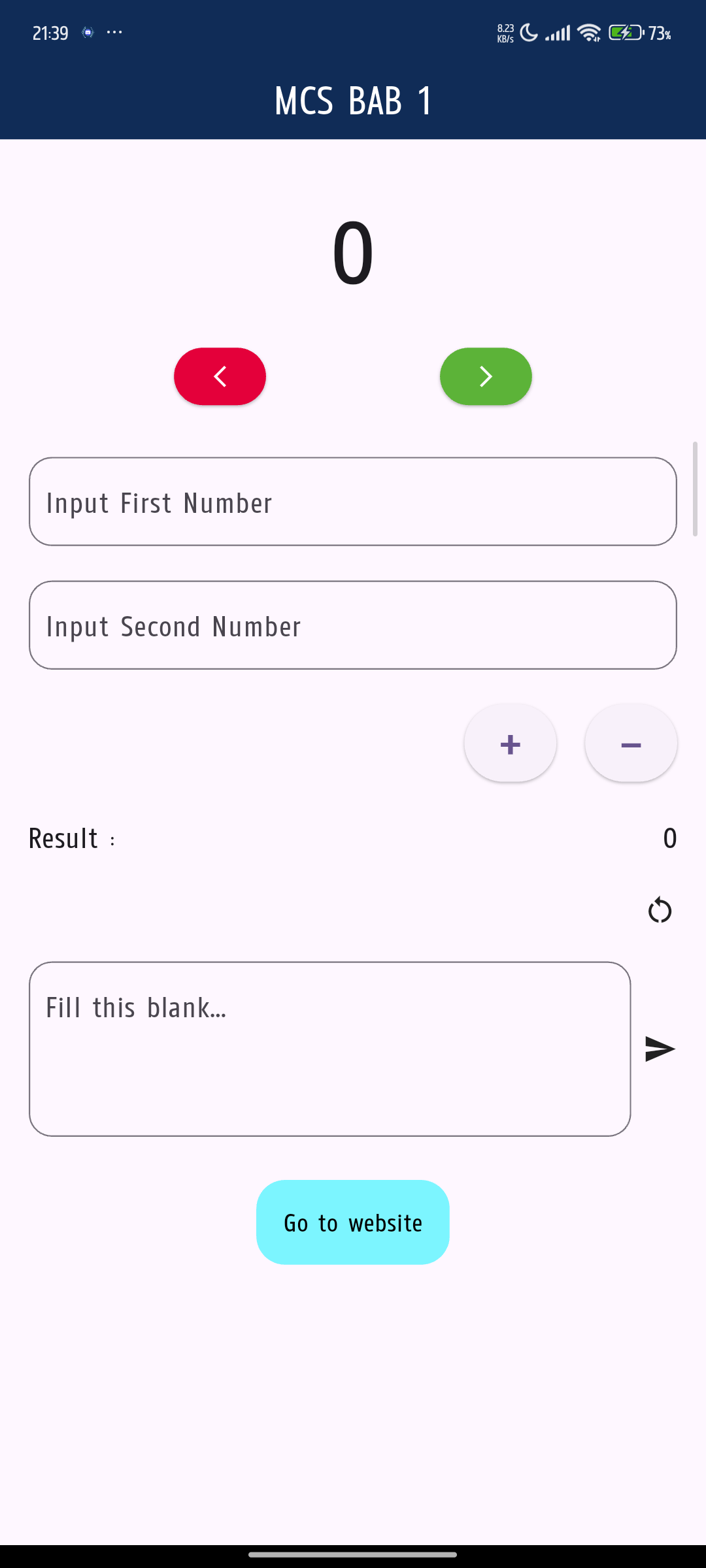
#### StatefulWidget():

1. Dapat menangani perubahan tampilan, Misal terdapat angka 1 pada halaman kemudian ingin dilakukakn perubahan tertentu dengan aktivitas tertentu pada angka tersebut, dengan StatefulWdget hal tersebut bisa dilakukan
2. Dibuat dengan extends dari StatefulWidget
3. Menggunakan 2 class berbeda, yakni widget dan *state*

## Prosedur Praktikum

### Tampilan Aplikasi

Berikut merupakan tampilan dari aplikasi yang akan dibentuk pada praktikum bab 1.

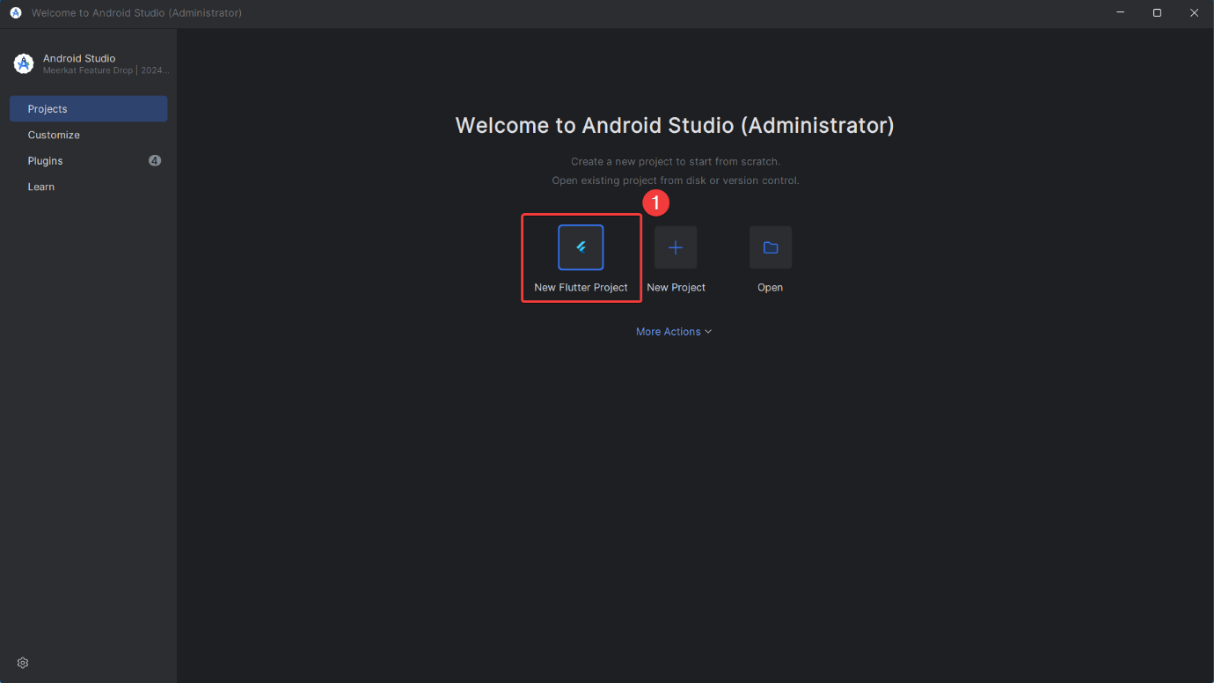
 

Gambar 1. Tampilan Halaman Aplikasi yang Akan diimplementasikan

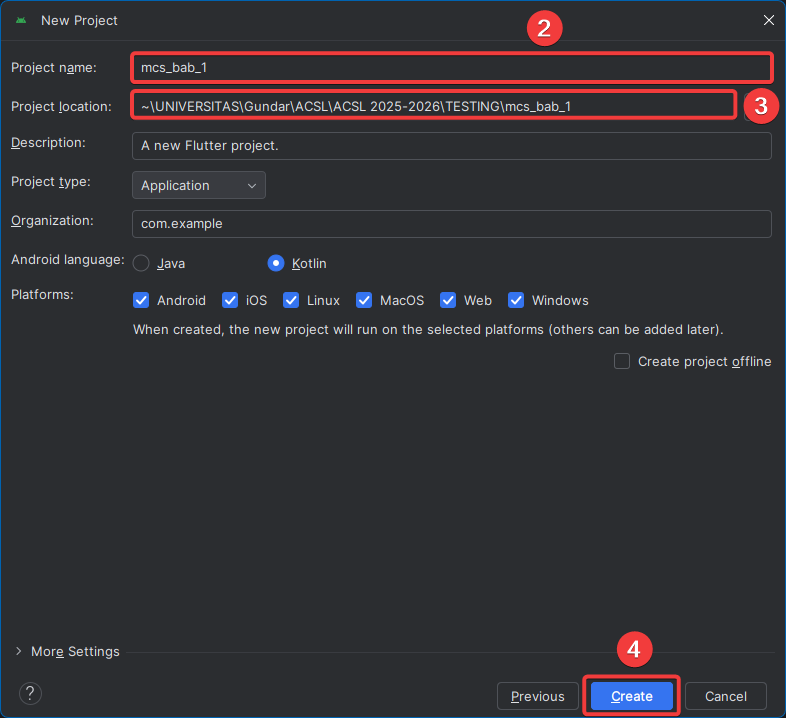
**Penjelasan terkait bagaimana cara aplikasi bekerja akan diterangkan oleh asisten yang mengajar.**

### Implementasi Aplikasi

Dalam mengimplementasikan tampilan dari desain aplikasi di atas, terdapat beberapa langkah yang harus dilewati terlebih dahulu agar proses praktikum dapat berjalan dengan lancar dan terselesaikan sesuai dengan apa yang dituju. Untuk membuat *project* flutter pada android studio, pilihlah menu **New Flutter Project** yang terdapat pada halaman utama android studio. Kemudian tentukanlah nama *project* dan tempat untuk menyimpan file *project* tersebut. **Pastikan nama file diketik dengan format *lowercase* dan jika nama file lebih dari 1 kata, maka pisahkan dengan format snake\_case.**

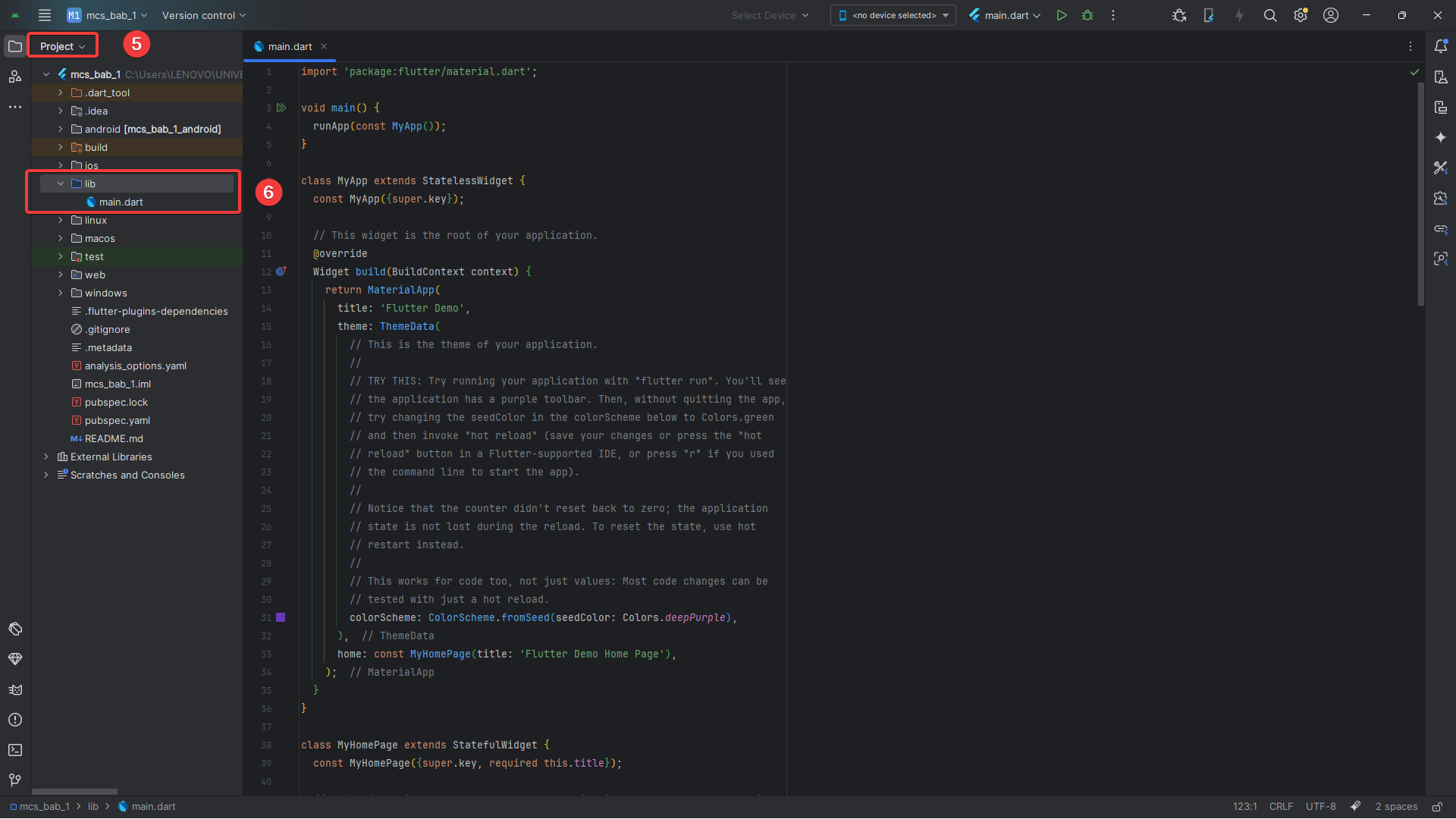


Gambar 1. Tampilan Awal *Software* Android *Studio*



Gambar 1. Proses Pembuatan *Project* Baru

Jika file *project* sudah berhasil terbentuk, pastikan tampilan awal android studio sudah berada pada menu ***project*** untuk mempermudah dalam pengerjaan suatu *project.* Setelah berada pada menu *project,* android studio akan menampilkan beberapa file konfigurasi yang digunakan dalam membangun *project* flutter, seperti android, ios, lib, linux, macos, dan file-file lainnya. Namun, untuk membentuk sebuah aplikasi dengan flutter folder yang difokuskan adalah folder **lib.**



Gambar 1. Tampilan *Default* Android *Studio*

Bukalah folder lib yang telah disediakan oleh flutter, maka akan terlihat 1 file dengan *exstension* dart bernama **main.dart** yang merupakan file utama sekaligus file yang akan dijalankan paling pertama saat program dijalankan. Ubahlah kode program *default* yang ada pada file tersebut dengan kode program berikut:

|  |
| --- |
| void main() {  runApp(const MyApp());  }  class MyApp extends StatelessWidget {  const MyApp({super.key});  @override  Widget build(BuildContext context) {  return MaterialApp(  title: 'MCS BAB 1',  debugShowCheckedModeBanner: false,  theme: ThemeData(  colorScheme: ColorScheme.fromSeed(seedColor: Colors.deepPurple),  useMaterial3: true,  ),  home: const HomePage(),  );  }  } |

Pada saat membangun *project* menggunakan flutter, fungsi **main()** merupakan fungsi utama yang wajib dihadirkan dan merupakan fungsi yang paling pertama dijalankan. Dalam fungsi ini, program akan memanggil fungsi **runApp()** yang akan menghubungkan kode program dart dengan flutter. Fungsi ini akan memanggil *class* **MyApp()** yang meng*extends* StetlessWidget. *Class* MyAppp() akan mengembalikan widget **MaterialApp()** yang di dalamnya berisikan beberapa properti yang digunakan, seperti **title:** yang akan memberikan label pada aplikasi, **debugShowCheckedModeBanne:** yang akan menerima nilai *boolean true or false* untuk mengatur kehadiran dari *banner debug* yang berada pada pojok kanan atas, **theme:** yang digunakan untuk konfigurasi tema aplikasi, dan **home:** yang akan mengatur navigasi halaman pada saat aplikasi dijalankan sekaligus menetapkan halaman pertama yang akan ditampilkan oleh aplikasi.

Setelah membangun fondasi awal dari aplikasi, langkah berikutnya yang dilakukan adalah membuat file baru bernama **home\_page.dart** yang akan membentuk halaman *home* aplikasi dengan menggunakan beberapa baris kode program. Berikut merupakan kode program yang digunakan dalam membentuk halaman *home* aplikasi tersebut:

|  |
| --- |
| class HomePage extends StatefulWidget {  const HomePage({super.key});  @override  State<HomePage> createState() => \_HomePageState();  }  class \_HomePageState extends State<HomePage> {  int start = 0;  int result = 0;  TextEditingController input1Controller = TextEditingController();  TextEditingController input2Controller = TextEditingController();  TextEditingController messageController = TextEditingController();  final Uri flutterUrl = Uri.parse("https://flutter.dev");  // FUNCTION FOR INCREMENT  void increment() {  setState(() {  start++;  });  }  // FUNCTION FOR DECREMENT  void decrement() {  setState(() {  start--;  if (start <= 0) {  start = 0;  }  });  }  // FUNCTION FOR SUMOF  void sumOf({required int x, required int y}) {  setState(() {  result = x + y;  });  }  // FUNCTION FOR SUBSTRACTION  void substractionFrom({required int x, required int y}) {  setState(() {  result = x - y;  });  }  // FUNTION FOR RESET FIELD 1, FIELD 2, NUMBER  void resetResult() {  setState(() {  input1Controller.text = "";  input2Controller.text = "";  result = 0;  });  }  @override  void dispose() {  input1Controller.dispose();  input2Controller.dispose();  messageController.dispose();  super.dispose();  }  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  backgroundColor: const Color(0xff102C57),  title: const Text("MCS BAB 1", style: TextStyle(color: Colors.white)),  centerTitle: true,  ),  body: ListView(  children: [  const SizedBox(height: 32),  Center(  child: Text(start.toString(), style: const TextStyle(fontSize: 50)),  ),  const SizedBox(height: 20),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  // BUTTON FOR DECREAS NUMBER  ElevatedButton(  onPressed: () {  decrement();  },  style: ElevatedButton.styleFrom(  backgroundColor: const Color(0xffE4003A),  ),  child: const Icon(  Icons.arrow\_back\_ios\_new,  color: Colors.white,  ),  ),  // BUTTON FOR INCREMENT NUMBER  ElevatedButton(  onPressed: () {  increment();  },  style: ElevatedButton.styleFrom(  backgroundColor: const Color(0xff5CB338),  ),  child: const Icon(Icons.arrow\_forward\_ios, color: Colors.white),  ),  ],  ),  const SizedBox(height: 32),  // FIELD FOR INPUT FIRST NUMBER  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: TextFormField(  controller: input1Controller,  keyboardType: TextInputType.number,  inputFormatters: <TextInputFormatter>[  FilteringTextInputFormatter.allow(RegExp(r'[0-9]')),  ],  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Input First Number",  ),  ),  ),  const SizedBox(height: 24),  // FIELD FOR INPUT SECOND NUMBER  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: TextFormField(  controller: input2Controller,  keyboardType: TextInputType.number,  inputFormatters: <TextInputFormatter>[  FilteringTextInputFormatter.allow(RegExp(r'[0-9]')),  ],  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Input Second Number",  ),),  ),  const SizedBox(height: 24),  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.end,  children: [  // BUTTON FOR SUM  ElevatedButton(  onPressed: () {  sumOf(  x: int.parse(input1Controller.text),  y: int.parse(input2Controller.text),  );  },  child: const Text(  "+",  style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold),  ),  ),  SizedBox(width: 20),  // BUTTON FOR SUBSTRACTION  ElevatedButton(  onPressed: () {  substractionFrom(  x: int.parse(input1Controller.text),  y: int.parse(input2Controller.text),  );  },  child: const Text(  "-",  style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold),  ),),  ],  ),  ),  const SizedBox(height: 24),  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  const Text(  "Result :",  style: TextStyle(fontSize: 16, fontWeight: FontWeight.w600),  ),  Text(  "$result",  style: const TextStyle(  fontSize: 16,  fontWeight: FontWeight.w700,  ),),  ],  ),  ),  const SizedBox(height: 24),  // BUTTON FOR RESET RESULT  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.end,  children: [  GestureDetector(  child: const Icon(Icons.restart\_alt),  onTap: () {  resetResult();  },  ),  ],  ),  ),  const SizedBox(height: 24),  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  children: [  // CONTAINER FOR CREATE MESSAGE  Expanded(  child: TextFormField(  controller: messageController,  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Fill this blank...",  ),  maxLines: 3,  ),  ),  const SizedBox(width: 8),  // BUTTON FOR SEND MESSAGE  GestureDetector(  child: const Icon(Icons.send),  onTap: () {  Navigator.push(  context,  MaterialPageRoute(  builder: (context) =>  ReceiverPage(message: messageController.text),  ),);  },),  ],),  ),  const SizedBox(height: 30),  Center(  child: GestureDetector(  child: Container(  padding: const EdgeInsets.symmetric(  vertical: 12,  horizontal: 14,  ),  decoration: BoxDecoration(  borderRadius: BorderRadius.circular(20),  color: const Color(0xff7CF5FF),  ),  child: Padding(  padding: const EdgeInsets.all(5),  child: Text(  "Go to website",  style: const TextStyle(color: Colors.black),  ),),  ),  onTap: () {  launchUrl(flutterUrl);  },  ),  ),  ],  ),);  }  } |

*Class* HomePage merupakan *class* pertama yang akan dipanggil pada saat argumen home: pada widget MaterialApp() dijalankan. *Class* ini akan meng*extends* statefulwidgetuntuk membagnun halamannya. Untuk membentuk halaman *home,* penulisan kode dibagi ke dalam beberapa bagian.

|  |
| --- |
| int start = 0;  int result = 0;  TextEditingController input1Controller = TextEditingController();  TextEditingController input2Controller = TextEditingController();  TextEditingController messageController = TextEditingController();  final Uri flutterUrl = Uri.parse("https://flutter.dev"); |

Bagian pertama yang dilakukan adalah mendefinisikan beberapa variabel yang nantinya akan digunakan untuk membangun halaman *home.* Pada pendefinisian tersebut, terdapat 2 variabel bertipe integer, yakni variabel start dan result yang memiliki nilai *default* 0. Kemudian terdapat 3 variabel bertipe TextEditingController dimana masing-masing variabel tersebut akan menyimpan objek *controller* dari TextEditingController() yang dapat digunakan untuk meminta dan mengambil *input* data dari pengguna. Kemudian, terdapat variabel flutterUrl yang berisikan sebuah url dari website *official* flutter. Variabel ini akan melakukan konversi terlebih dahulu dari bentuk String menjadi ke dalam bentuk objek uri yang dipahami oleh dart dengan menggunakan perintah **Uri.parse().**

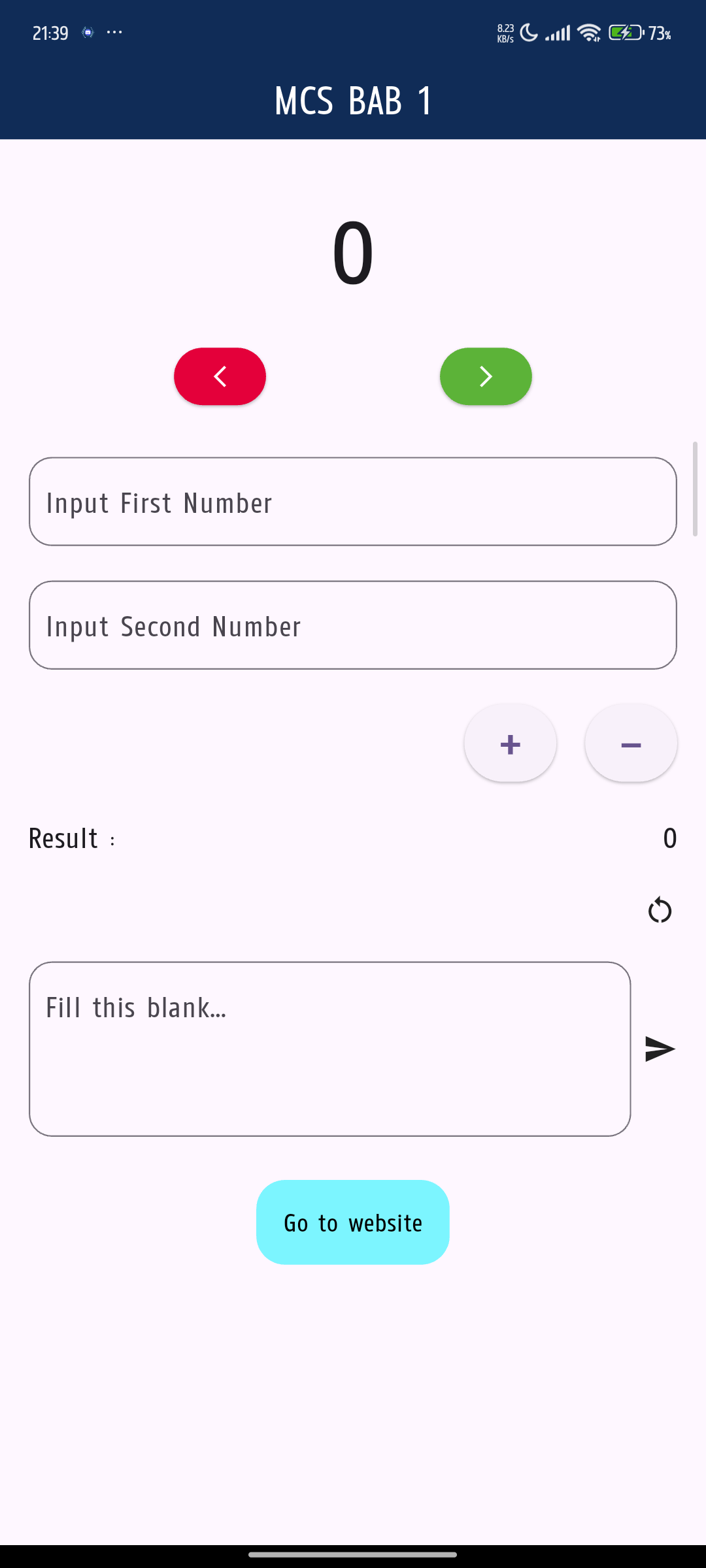
Setelah mendefinisikan beberapa variabel yang akan digunakan untuk membangun halaman *home,* langkah selanjutnya yang dilakukan adalah membuat beberapa fungsi yang nantinya akan digunakan juga pada halaman *home.* Terdapat beberapa fungsi yang akan dibangun, seperti fungsi **increment(), decrement(), sumOf(), substractionFrom(), resetResult(),** dan **dispose().**

|  |
| --- |
| // FUNCTION FOR INCREMENT  void increment() {  setState(() {  start++;  });  }  // FUNCTION FOR DECREMENT  void decrement() {  setState(() {  start--;  if (start <= 0) {  start = 0;  }  });  }  // FUNCTION FOR SUMOF  void sumOf({required int x, required int y}) {  setState(() {  result = x + y;  });  }  // FUNCTION FOR SUBSTRACTION  void substractionFrom({required int x, required int y}) {  setState(() {  result = x - y;  });  }  // FUNTION FOR RESET FIELD 1, FIELD 2, NUMBER  void resetResult() {  setState(() {  input1Controller.text = "";  input2Controller.text = "";  result = 0;  });  }  @override  void dispose() {  input1Controller.dispose();  input2Controller.dispose();  messageController.dispose();  super.dispose();  } |

Fungsi **increment()** berisikan beberapa baris program yang akan meng*handle button* **arrow\_forward\_ios** dalam menambahkan nilai dari variabel *start.* Sedangkan fungsi **decrement()** akan berisikan beberapa baris program yang akan meng*handle* pengurangan nilai dari variabel *start* dan diberikan sedikit pengkondisian, dimana ketika variabel *start* bernilai 0, maka fungsi ini tidak akan mengurangi nilai pada variabel *start* dan berhenti pada nilai 0.

Fungsi **sumOf()** berisikan beberapa baris program yang akan menambahkan 2 nilai yang di*input* oleh pengguna ke dalam 2 *field* berbeda. Sdangkan, fungsi **substractionFrom()** berisikan baris kode program yang akan mengurangi 2 nilai yang telah di*input* oleh pengguna. Nilai yang dihasilkan dari fungsi sumOf() dan substractionFrom() akan disimpan dalam variabel result. Kemudian fungsi **resetResult()** berisikan beberapa baris program yang akan mengosongkan kembali nilai pada variabel input1Controller, input2Controller, dan mereset kembali nilai dari variabel *start.* Fungsi **dispose()** digunakan untuk mencegah terjadinya kebocoran memori yang dapat disebabkan oleh *controller.*

Setelah mendefinisikan beberapa variabel dan fungsi yang akan digunakan, langkah berikutnhya yang dilakukan adalah membangun tampilan dari halaman *home.* Untuk membangun halaman dari sebuah aplikasi menggunkaan flutter, diperlukan pengembalian terhadap widget **Scaffold().** Berbeda dengan widget MaterialApp() yang hanya berisikan beberapa properti untuk konfigurasi aplikasi, widget Scaffold() digunakan untuk membangun tampilan aplikasi yang dipisahkan dalam properti **appBar:** dan **body:**

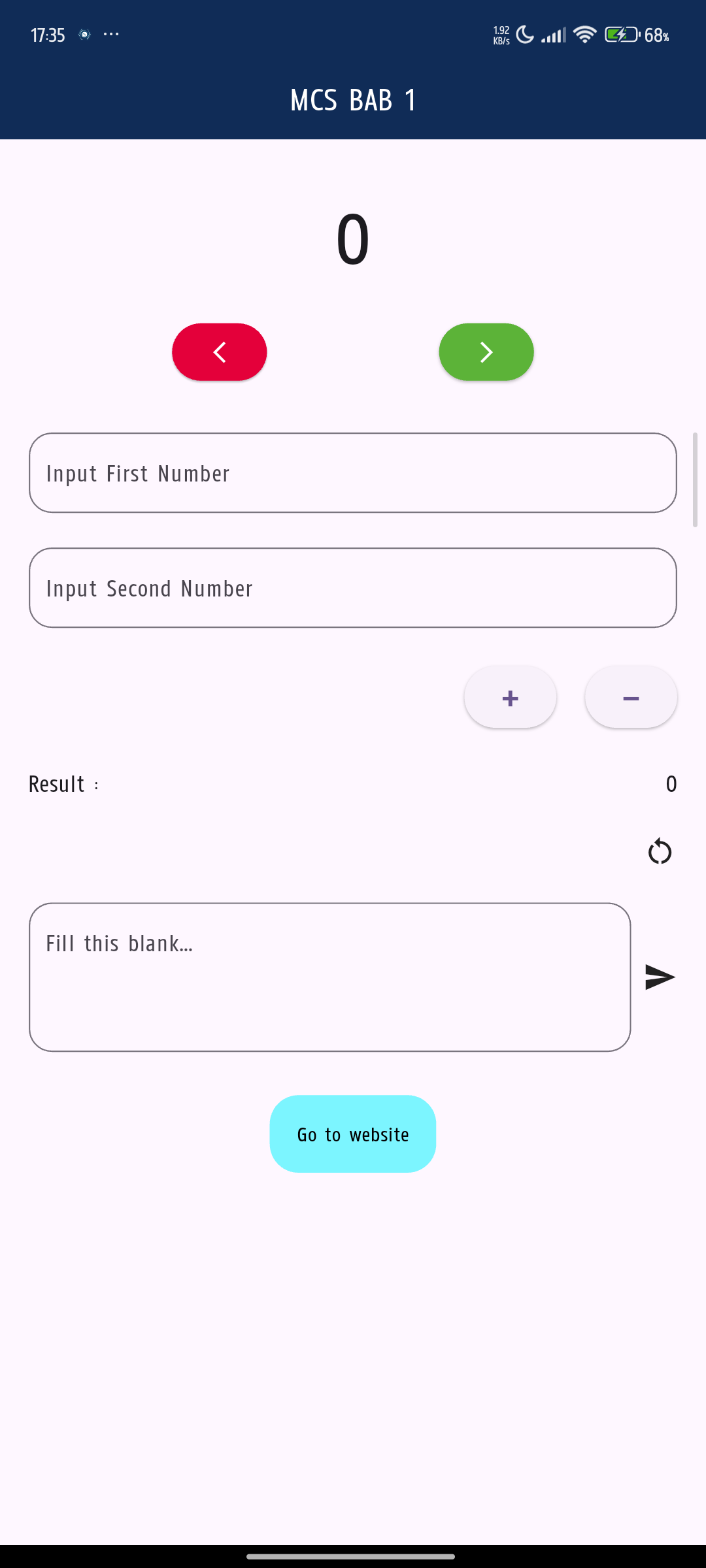


Gambar 1. Tampilan AppBar Aplikasi

|  |
| --- |
| return Scaffold(  appBar: AppBar(  backgroundColor: const Color(0xff102C57),  title: const Text("MCS BAB 1", style: TextStyle(color: Colors.white)),  centerTitle: true,  ),  // ...  ); |

Tampilan appbar yang dibentuk pada praktikum ini hanyalah tampilan appbar sederhana yang bertuliskan MCS BAB 1 yang berada di bagian tengah. Selain itu, appbar pada aplikasi ini diberikan *styling* warna dengan kode hexa 0xff102C57. Kode hexa dari warna yang lainnya dapat dicari pada website [**https://colorhunt.co**](https://colorhunt.co)**.**

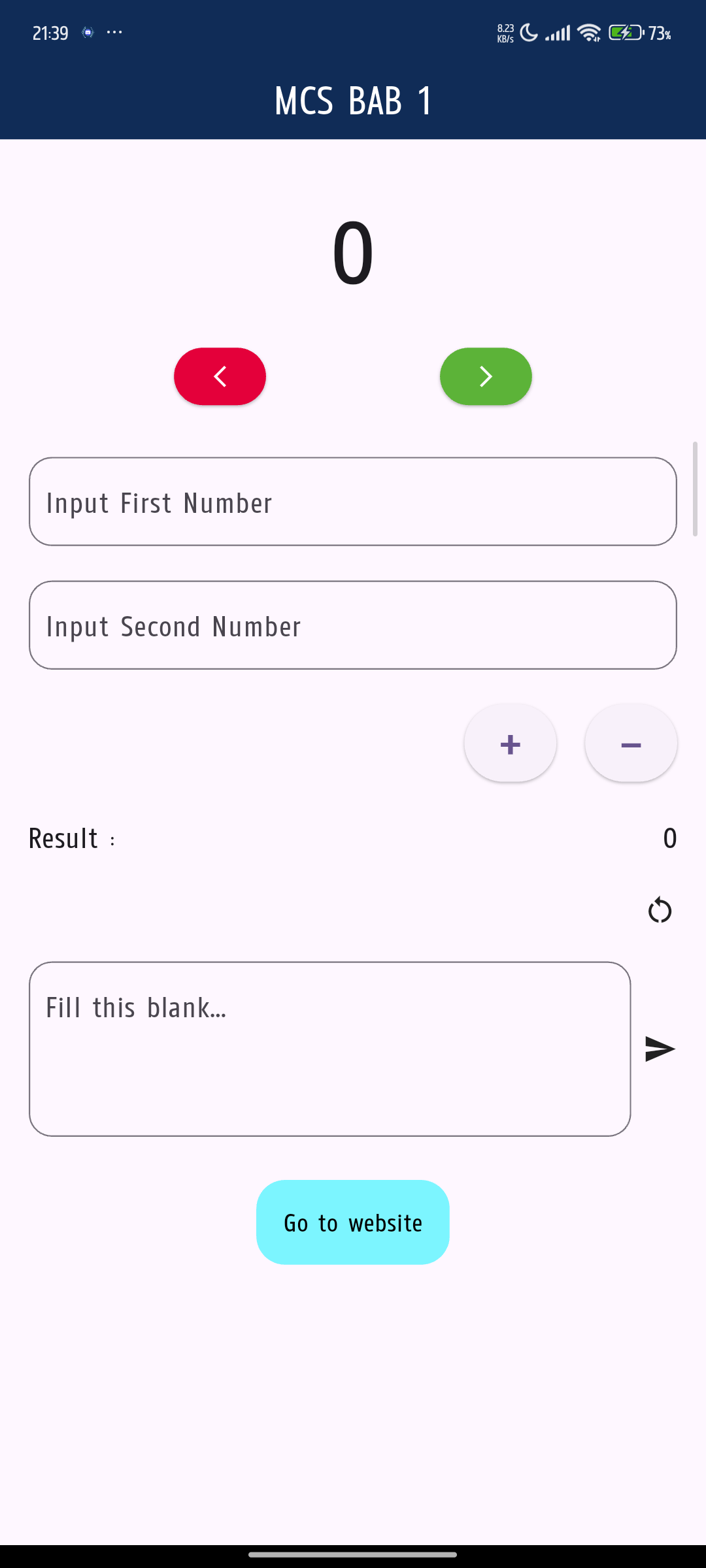
Selanjutnya properti **body:** digunkan untuk membangun bagian badan dari halaman aplikasi. Properti ini dibangun dengan menggunakan widget **ListView()** yang dapat memungkinkan pengguna untuk melakukan *scroll* terhadap aplikasi. Widget ini mengembalikan properti children: yang dapat menampung banyak widget di dalamnya.



Gambar 1. Area untuk Menambahkan Angka

|  |
| --- |
| return Scaffold(  // ...  body: ListView(  children: [  const SizedBox(height: 32),  Center(  child: Text(start.toString(), style: const TextStyle(fontSize: 50)),  ),  const SizedBox(height: 20),  Row(  mainAxisAlignment: MainAxisAlignment.spaceEvenly,  children: [  // BUTTON FOR DECREAS NUMBER  ElevatedButton(  onPressed: () {  decrement();  },  style: ElevatedButton.styleFrom(  backgroundColor: const Color(0xffE4003A),  ),  child: const Icon(  Icons.arrow\_back\_ios\_new,  color: Colors.white,  ),  ),  // BUTTON FOR INCREMENT NUMBER  ElevatedButton(  onPressed: () {  increment();  },  style: ElevatedButton.styleFrom(  backgroundColor: const Color(0xff5CB338),  ),  child: const Icon(Icons.arrow\_forward\_ios, color: Colors.white),  ),  ],  ),  // ...  ); |

Widget **SizedBox()** digunakan untuk memberikan jarak antar widget. Sizedbox() memiliki properti **height:** yang akan memberikan jarak secara *vertical* dan properti **width:** yang akan memberikan jarak secara *horizontal.* Kemudian widget **Center()** digunakan untuk membuat seluruh widget yang ada di dalamnya berada pada bagian tengah layar aplikasi. Widget ini memiliki properti child: yang memanggil widget **Text()** untuk mengembalikan nilai yang tersimpan pada variabel *start*. Nilai awal yang ditampilkan pada bagian ini adalah 0 yang kemudian nanti akan berubah sesuai dengan *button* yang ditekan. Widget **Row()** berfungsi untuk membuat seluruh widget yang ada di dalamnya disusun secara *horizontal.* Sama seperti widget ListView(), widget ini juga dapat menampung banyak widget di dalamnya. Dalam widget ini, terdapat 2 widget **ElevatedButton()** yang digunakan untuk mengontrol perubahan angka *start* yang tertera di atasnya. ElevatedButton() pertama diberikan *styling* berupa *icon* panah ke kiri dengan warna *hexacode* 0xffE4003A. Properti onPressed: pada *button* ini diberikan fungsi decrement() untuk menurunkan angka Sedangkan, ElevatedButton() digunakan untuk menaikkan angka dengan pemanggilan fungsi increment() pada properti onPressed: dengan *styling icon* panah ke kanan dan dengan menggunakan warna *hexacode* 0xff5CB338.



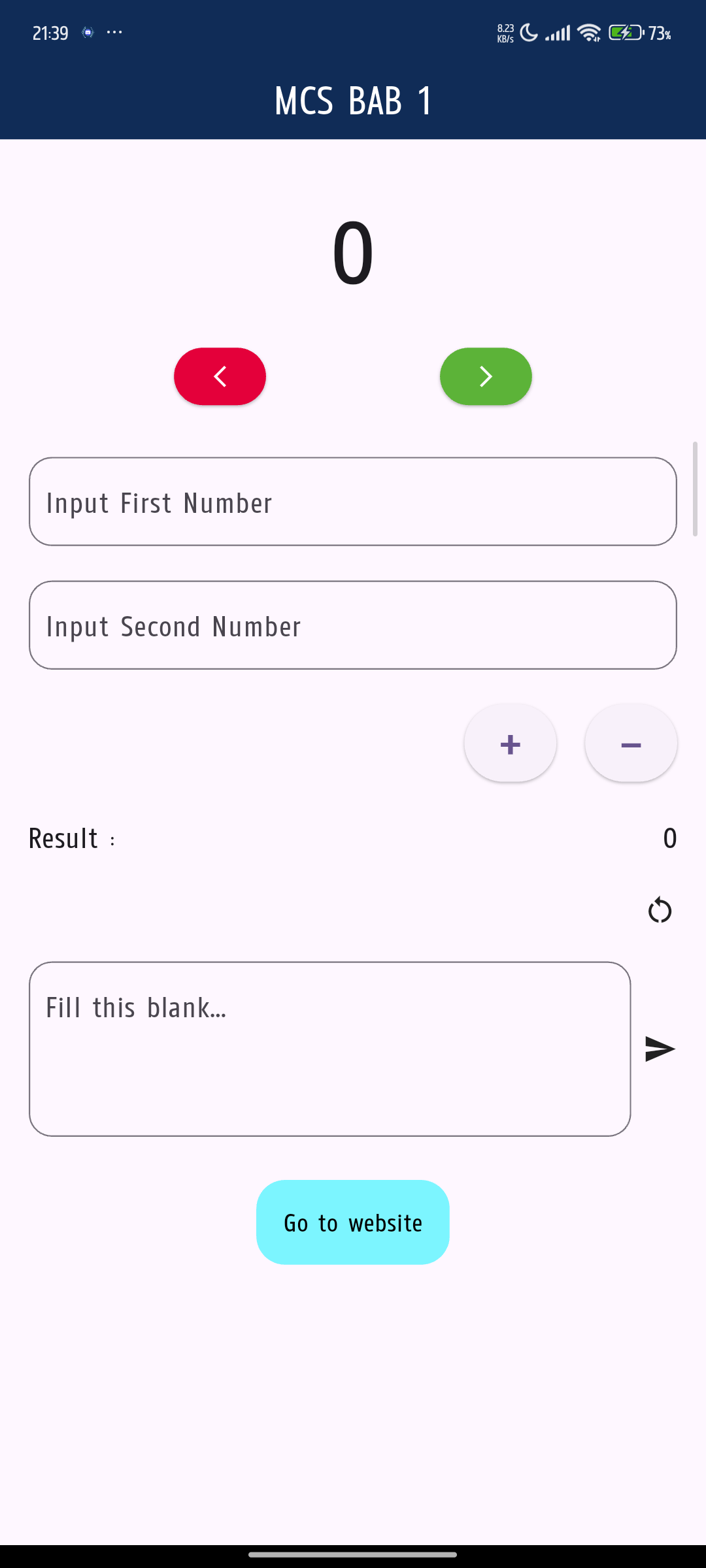
Gambar 1. Area untuk Menjumlahkan Angka

|  |
| --- |
| return Scaffold(  // ...  body: ListView(  children: [  // ...  // FIELD FOR INPUT FIRST NUMBER  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: TextFormField(  controller: input1Controller,  keyboardType: TextInputType.number,  inputFormatters: <TextInputFormatter>[  FilteringTextInputFormatter.allow(RegExp(r'[0-9]')),  ],  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Input First Number",  ),  ),  ),  const SizedBox(height: 24),  // FIELD FOR INPUT SECOND NUMBER  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: TextFormField(  controller: input2Controller,  keyboardType: TextInputType.number,  inputFormatters: <TextInputFormatter>[  FilteringTextInputFormatter.allow(RegExp(r'[0-9]')),  ],  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Input Second Number",  ),  ),  ),  const SizedBox(height: 24),  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.end,  children: [  // BUTTON FOR SUM  ElevatedButton(  onPressed: () {  sumOf(  x: int.parse(input1Controller.text),  y: int.parse(input2Controller.text),  );  },  child: const Text(  "+",  style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold),  ),  ),  SizedBox(width: 20),  // BUTTON FOR SUBSTRACTION  ElevatedButton(  onPressed: () {  substractionFrom(  x: int.parse(input1Controller.text),  y: int.parse(input2Controller.text),  );  },  child: const Text(  "-",  style: TextStyle(fontSize: 30, fontWeight: FontWeight.bold),  ),  ),  ],  ),  ),  const SizedBox(height: 24),  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.spaceBetween,  children: [  const Text(  "Result :",  style: TextStyle(fontSize: 16, fontWeight: FontWeight.w600),  ),  Text(  "$result",  style: const TextStyle(  fontSize: 16,  fontWeight: FontWeight.w700,  ),  ),  ],  ),  ),  const SizedBox(height: 24),  // BUTTON FOR RESET RESULT  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  mainAxisAlignment: MainAxisAlignment.end,  children: [  GestureDetector(  child: const Icon(Icons.restart\_alt),  onTap: () {  resetResult();  },  ),  ],  ),  ),  // ...  ); |

Kemudian di bawah 2 ElevatedButton tersebut, terdapat 2 widget **TextFormField()** yang berfungsi untuk menerima nilai *input* berupa angka yang akan dimasukkan oleh pengguna. Kedua widget tersebut hanya dapat menerima nilai *input* berupa angka yang berada dalam *range* 0 – 9 dengan radius setiap sudutnya sebesar 16. Pada masing-masing TextFormField() terdapat properti *controller* yang berfungsi sebagai penanda agar ketika pengguna meng*input* nilai pada satu *controller,* nilai *input* tersebut hanya masuk ke dalam *controller* yang diakses. Untuk membedakannya, properti *controller* pada TextFormField() pertama diambil dari variabel **input1Controller.** Sedangkan pada TexFormField() kedua diambil dari variabel **input2Controller.**

Berikutnya, di bawah widget TextFormField() kedua terdapat pemanggilan terhadap widget Row() yang di dalamnya menampung 2 widget ElevatedButton(). Widget ElevatedButton() pertama berfungsi untuk menjumlahkan 2 nilai yang di*input* pada 2 TextFormField() sebelumnya. Widget ini akan memanggil fungsi sumOf() yang berfungsi untuk menjumlahkan 2 angka yang telah dimasukkan pada input1Controller dan input2Controller. Sedangkan pada ElevatedButton() kedua digunakan untuk mengurangi 2 angka yang telah dimasukkan pada input1Controller dan input2Controller dengan pemanggilan terhadap fungsi substractionFrom(). Fungsi sumOf() dan substractionFrom() memiliki 2 konstruktor, yakni argumen x dan y yang bertipe data *integer* sehingga input1Controller dan input2Controller harus dikonversikan terlebih dahulu ke dalam bentuk *integer*.

Hasil penjumlahan dan pengurangan tersebut akan ditampung di dalam variabel *result* dan akan ditampilkan kepada pengguna dalam bentuk text. Dibawah area tersebut, terdapat *icon* ***restart*** yang berbentuk melingkar yang dibungkus oleh **GestureDetector()**. Widgetini berguna untuk membuat setiap widget yang diturunkannya dapat memiliki sebuah aksi ketika pengguna berinteraksi. *Icon* tersebut berfungsi untuk mengosongkan kembali *input1Controller*, *input2Controller* dan *result*.



Gambar 1. Area untuk Mengirim Pesan

|  |
| --- |
| return Scaffold(  // ...  body: ListView(  children: [  // ...  Container(  margin: const EdgeInsets.symmetric(horizontal: 20),  child: Row(  children: [  // CONTAINER FOR CREATE MESSAGE  Expanded(  child: TextFormField(  controller: messageController,  decoration: InputDecoration(  border: OutlineInputBorder(  borderRadius: BorderRadius.circular(16),  ),  hintText: "Fill this blank...",  ),  maxLines: 3,  ),  ),  const SizedBox(width: 8),  // BUTTON FOR SEND MESSAGE  GestureDetector(  child: const Icon(Icons.send),  onTap: () {  Navigator.push(  context,  MaterialPageRoute(  builder: (context) =>  ReceiverPage(message: messageController.text),  ),  );  },),  ],),  ),  const SizedBox(height: 30),  Center(  child: GestureDetector(  child: Container(  padding: const EdgeInsets.symmetric(  vertical: 12, horizontal: 14,  ),  decoration: BoxDecoration(  borderRadius: BorderRadius.circular(20),  color: const Color(0xff7CF5FF),  ),  child: Padding(  padding: const EdgeInsets.all(5),  child: Text(  "Go to website",  style: const TextStyle(color: Colors.black),  ),),  ),  onTap: () {  launchUrl(flutterUrl);  },  ),),  ],),  ); |

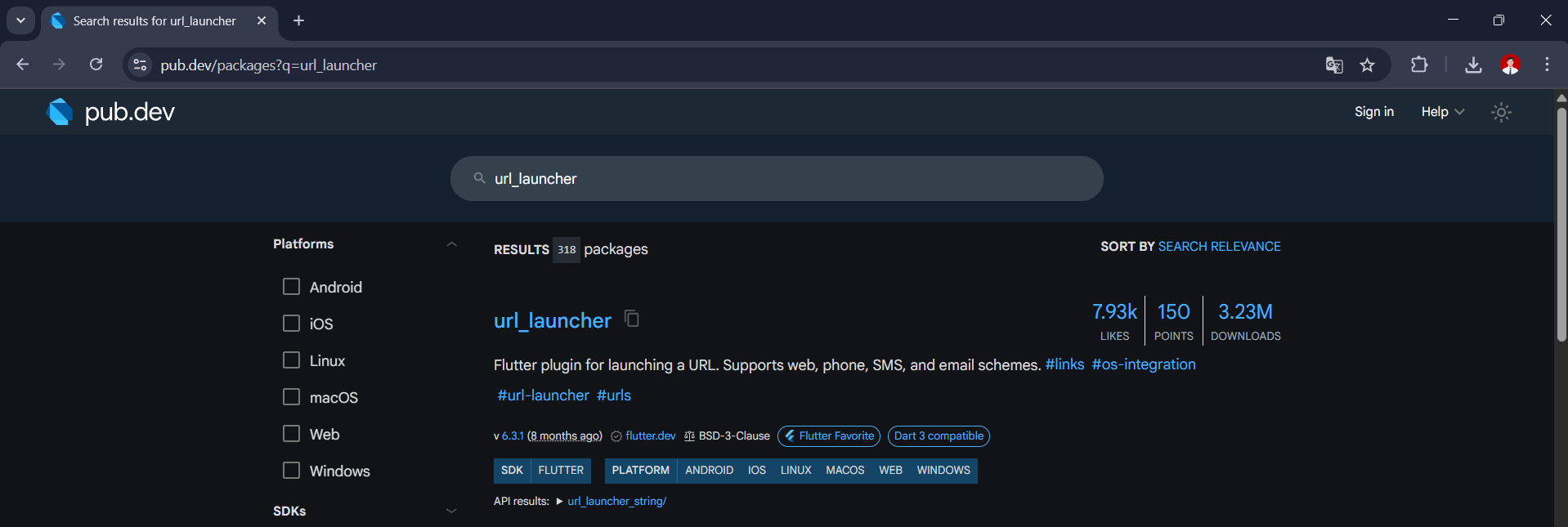
Berikutnya terdapat *field* besar yang juga akan menerima *input* dari pengguna berupa teks panjang yang nantinya dapat dikirimkan ke halaman selanjutnya, yakni halaman reveiver page. Namun, sebelum membangun area ini, diperlukan sebuah file bernama **receiver\_page.dart** yang akan menerima dan menampilkan pesan yang dituliskan oleh pengguna pada field tersebut. Berikut merupakan kode yang digunakan pada file receiver\_page.dart:

|  |
| --- |
| class ReceiverPage extends StatefulWidget {  String message;  ReceiverPage({super.key, required this.message});  @override  State<ReceiverPage> createState() => \_ReceiverPageState();  }  class \_ReceiverPageState extends State<ReceiverPage> {  @override  Widget build(BuildContext context) {  return Scaffold(  appBar: AppBar(  title: Text("Message Page"),  centerTitle: true  ),  body: Center(  child: Text(widget.message, style: TextStyle(fontSize: 30))  ),);  }  } |

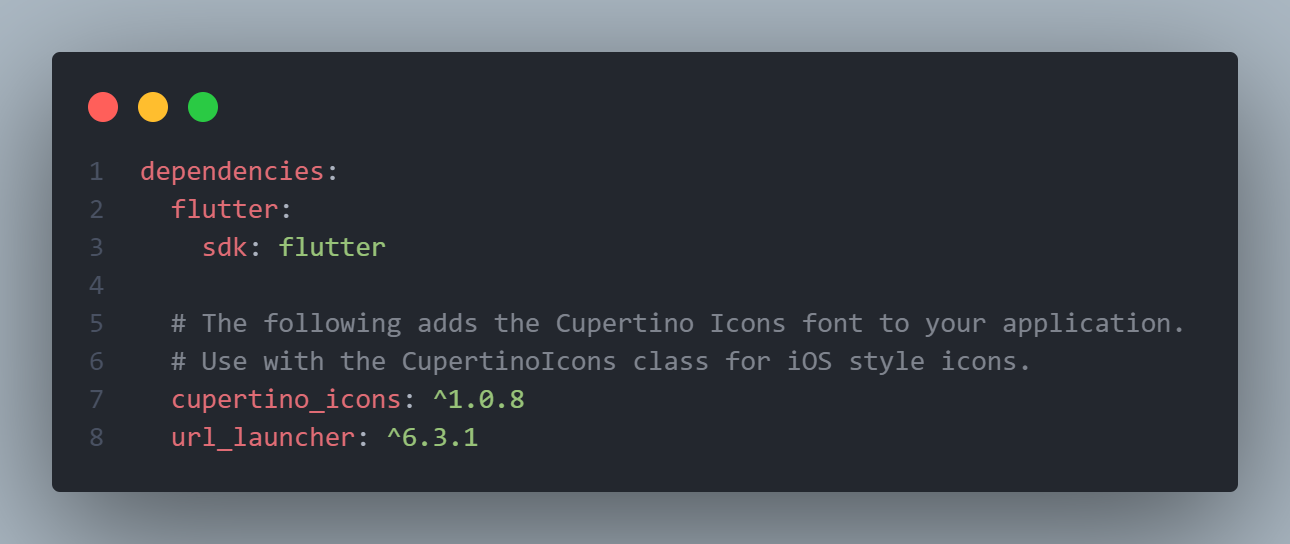
*Class* ReceiverPage() memiliki sebuah parameter yang diambil dari variabel message. Hal tersebut digunakan untuk menampung nilai *input* yang dimasukkan di *class* sebelumnya. Hasil dari *input* pesantersebut nantinya akan ditampilkan pada bagian tengah halaman aplikasi dengan penggunaan dari widget Center().

Setelah membuat file receiver\_page.dart dan menuliskan kode di atas, kembalilah ke file home\_page.dart dan lanjutkan penulisan kode pada bagian TextFormField() yang berada di bawah area *button restart.* Area tersebut di bungkus dalam widget Row() yang di dalamnya terdapat widget TextFormField() yang digunakan untuk menerima nilai *input* dari pengguna. TextFormField() tersebut memiliki *styling* terhadap setiap ujung dari bordernya dengan ukuran 16 dan terdapat penggunaan properti **hintText:** yang akan memberikan kalimat petunjuk terhadap *field* tersebut. Kemudian widget selanjutnya yang digunakan adalah widget *icon* yang dibungkus dengan widget GestureDetector(). Widget ini akan membuat *icon send* yang ada di sebelah TextFormField dapat disentuh dan akan membuat tampilan aplikasi mengarah pada halaman yang lain, yakni halaman ReceiverPage dengan membawa pesan yang telah di *input* oleh pengguna ke dalam field.

Berikutnya pada bagian akhir dari halaman tersebut, terdapat sebuah *button* yang ketika ditekan akan mengarahkan pengguna ke halaman website *official* flutter. Untuk membuat agar aplikasi dapat membuka url tersebut, perlu dilakukan penambahan terhadap sebuah *package* bernama **url\_launcher**ke dalam file **pubspec.yaml.** *Package* tersebut ditambahkan pada bagian **dependencies** dengan indentasi yang setara dengan cupertino\_icons. *Packages* tersebut dapat dicari pada halaman website resmi yang disediakan oleh flutter, yakni [**https://pub.dev/**](https://pub.dev/). Website tersebut menyediakan seluruh *packages* yang dapat digunakan untuk *project* Flutter.



Gambar 1. Tampilan Halaman *Website* Pub.dev



Gambar 1. *Package* yang digunakan