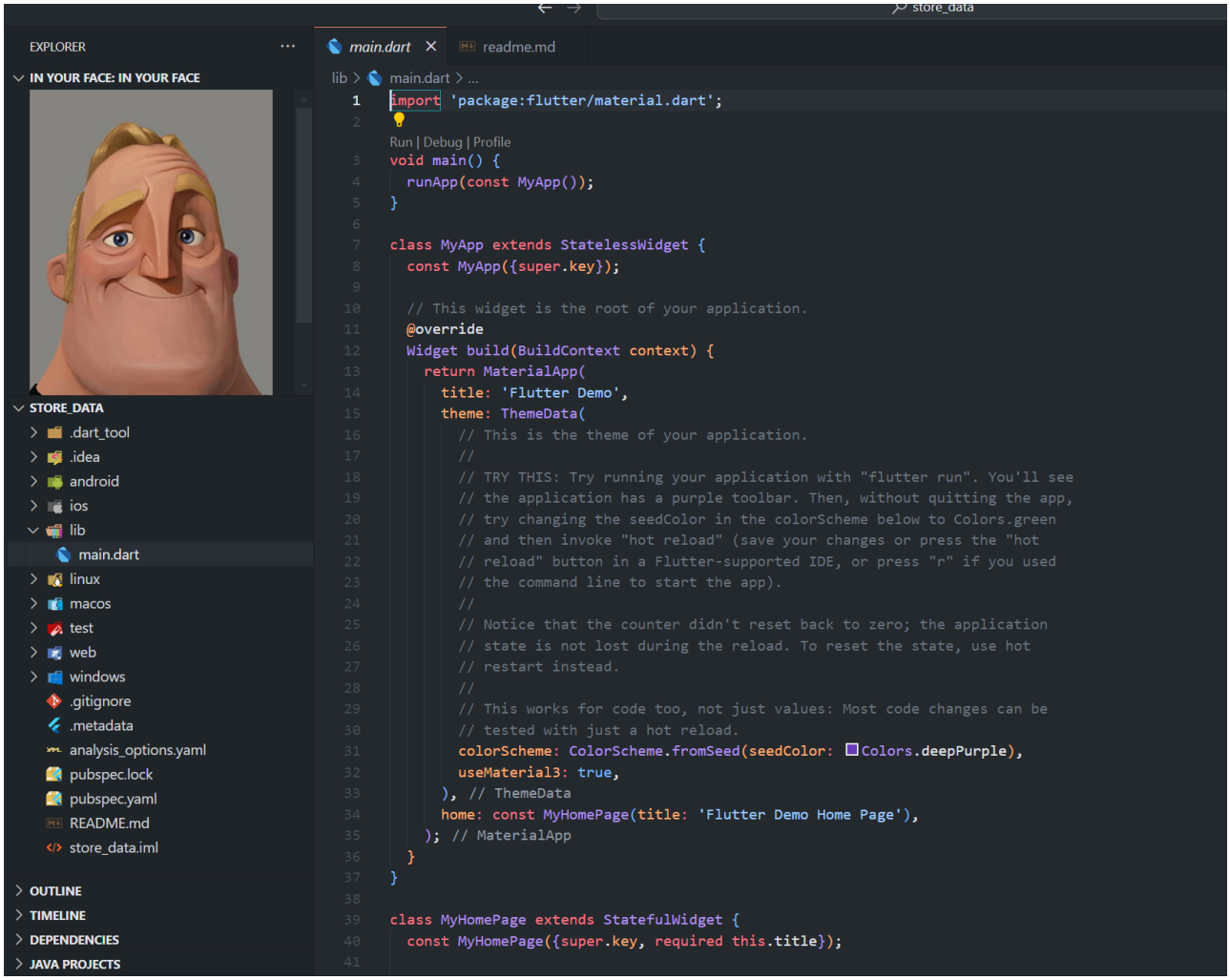
pertemuan 14

Praktikum 1

1. Di editor favorit Anda, buat proyek Flutter baru dan beri nama store_data



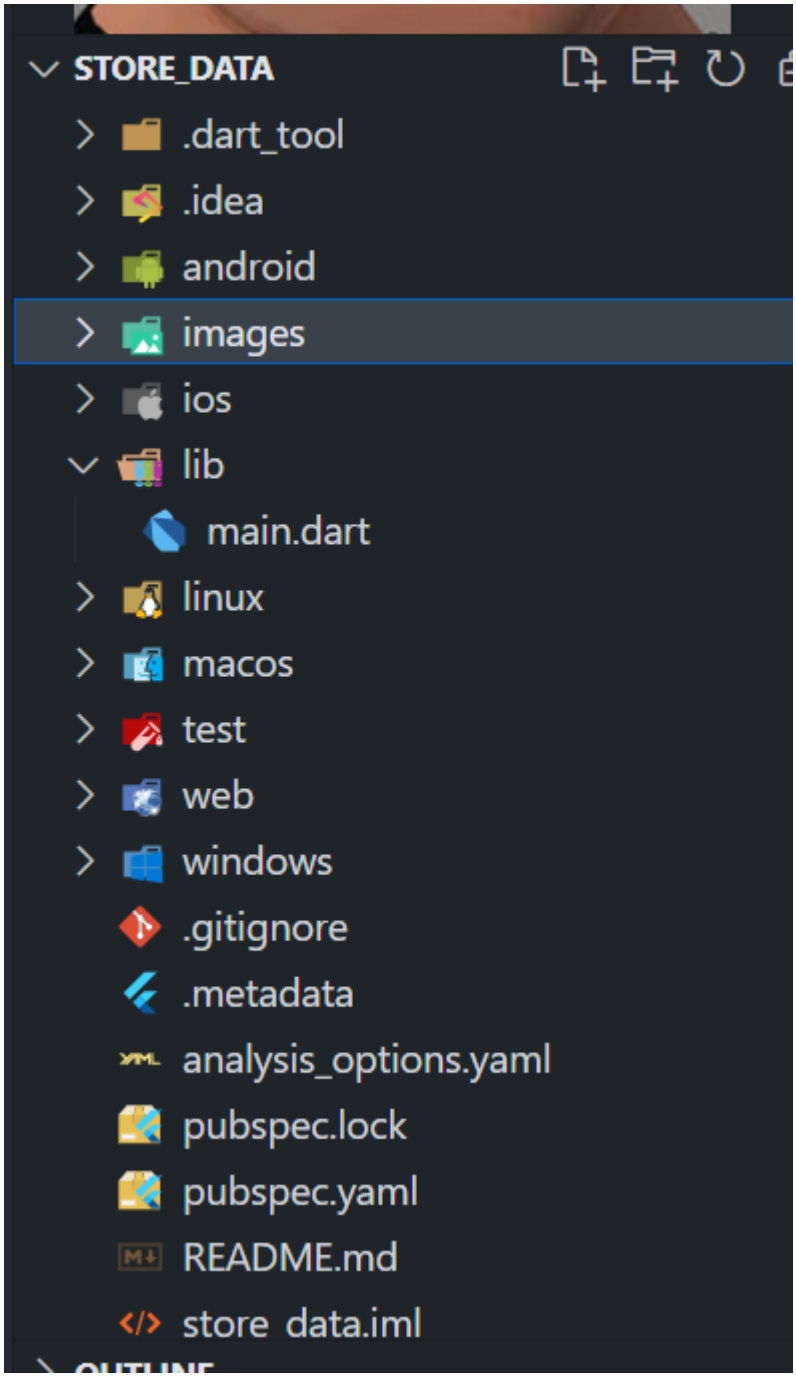
2. Pada file main.dart, hapus kode yang ada dan tambahkan kode awal untuk aplikasi dengan kode berikut:

main.dart readme.md

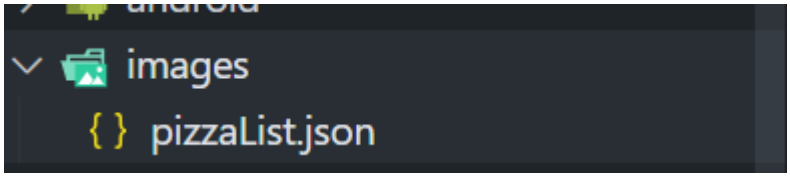
lib > main.dart > ...

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

Run | Debug | Profile
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter JSON Demo',
      theme: ThemeData(primarySwatch: Colors.blue),
      home: MyHomePage(),    Use 'const' with the constructor to impr
    ); // MaterialApp
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('JSON'),
      ), // AppBar
      body: Container(),
    ); // Scaffold
  }
}
```

3. Tambahkan folder baru ke root proyek anda dengan nama assets



4. Di dalam folder aset, buat file baru bernama pizzalist.json dan salin konten yang tersedia di tautan https://gist.github.com/simoales/a33c1c2abe78b48a75ccfd5fa0de0620. File ini berisi daftar objek JSON.



5. Di file pubspec.yaml, tambahkan referensi ke folder aset baru, seperti yang ditunjukkan di sini:

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

6. Pada kelas _MyHomePageState, di main.dart, tambahkan sebuah variabel state bernama pizzaString:

```
28
29      String pizzaString = '';
30
```

7. Untuk membaca isi file pizzalist.json, di bagian bawah kelas _MyHomePageState di main.dart, tambahkan metode asinkron baru yang disebut readJsonFile, yang akan mengatur nilai pizzaString, seperti yang ditunjukkan di sini:

```
Future readJsonFile() async{
  String myString = await DefaultAssetBundle.of(context).loadString('i
  setState(() {
    pizzaString = myString;
  });
}
```

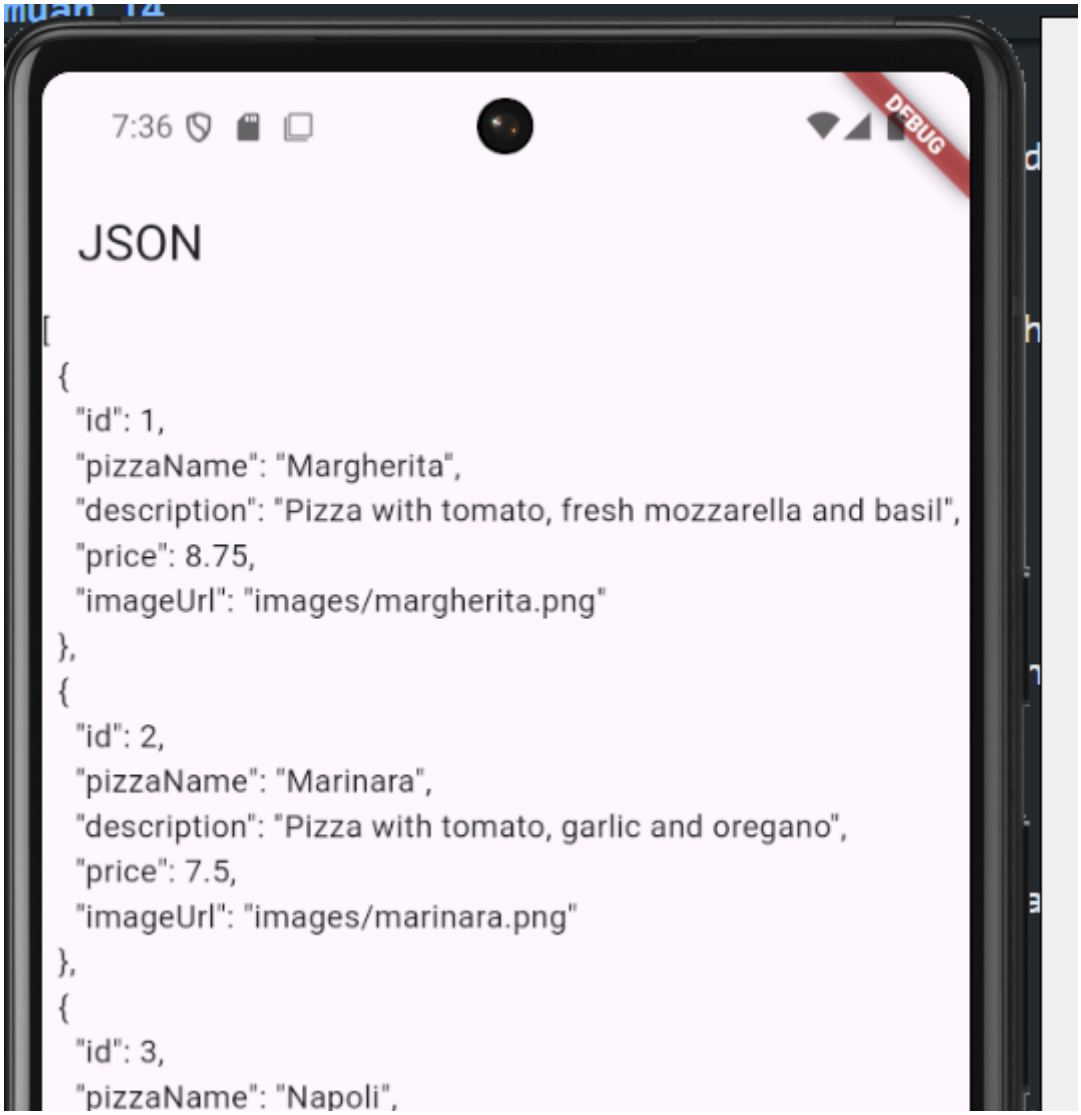
8. Pada kelas _MyHomePageState, timpa metode initState dan, di dalamnya, panggil metode readJsonFile:

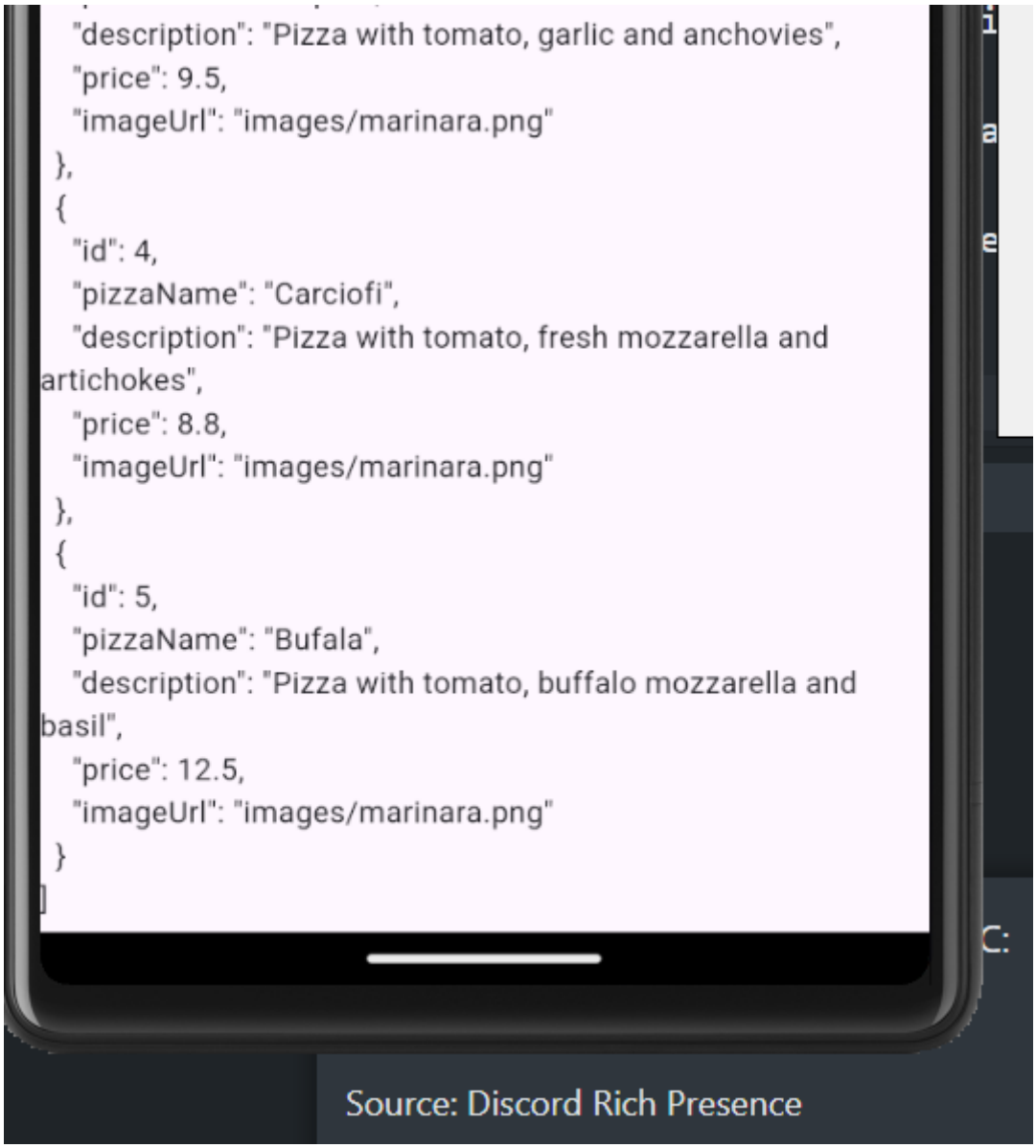
```
@override
void initState() {
  super.initState();
  readJsonFile();
}
```

9. Sekarang, kita ingin menampilkan JSON yang diambil di properti dalam Scaffold. Untuk melakukannya, tambahkan widget Teks sebagai child dari Container kita:

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('JSON'),
    ), // AppBar
    body: Center(
      child: Text(pizzaString),
    ), // Center
  ); // Scaffold
}
}
```

10. Mari kita jalankan aplikasinya. Jika semuanya berjalan seperti yang diharapkan, Anda akan melihat konten file JSON di layar





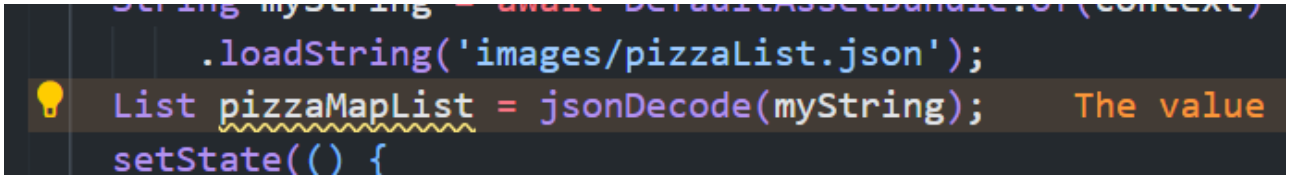
11. Kita ingin mengubah String ini menjadi sebuah List of Objects. Kita akan mulai dengan membuat kelas baru. Dalam folder lib aplikasi kita, buat file baru bernama pizza.dart.

12. Di dalam file tersebut, tentukan properti kelas Pizza:

```dart
class Pizza{
  final int id;
  final String pizzaName;
  final double price;
  final String imageUrl;

  const Pizza({required this.id, required this.pizzaName, required this.price, required this.imageUrl});
}
```

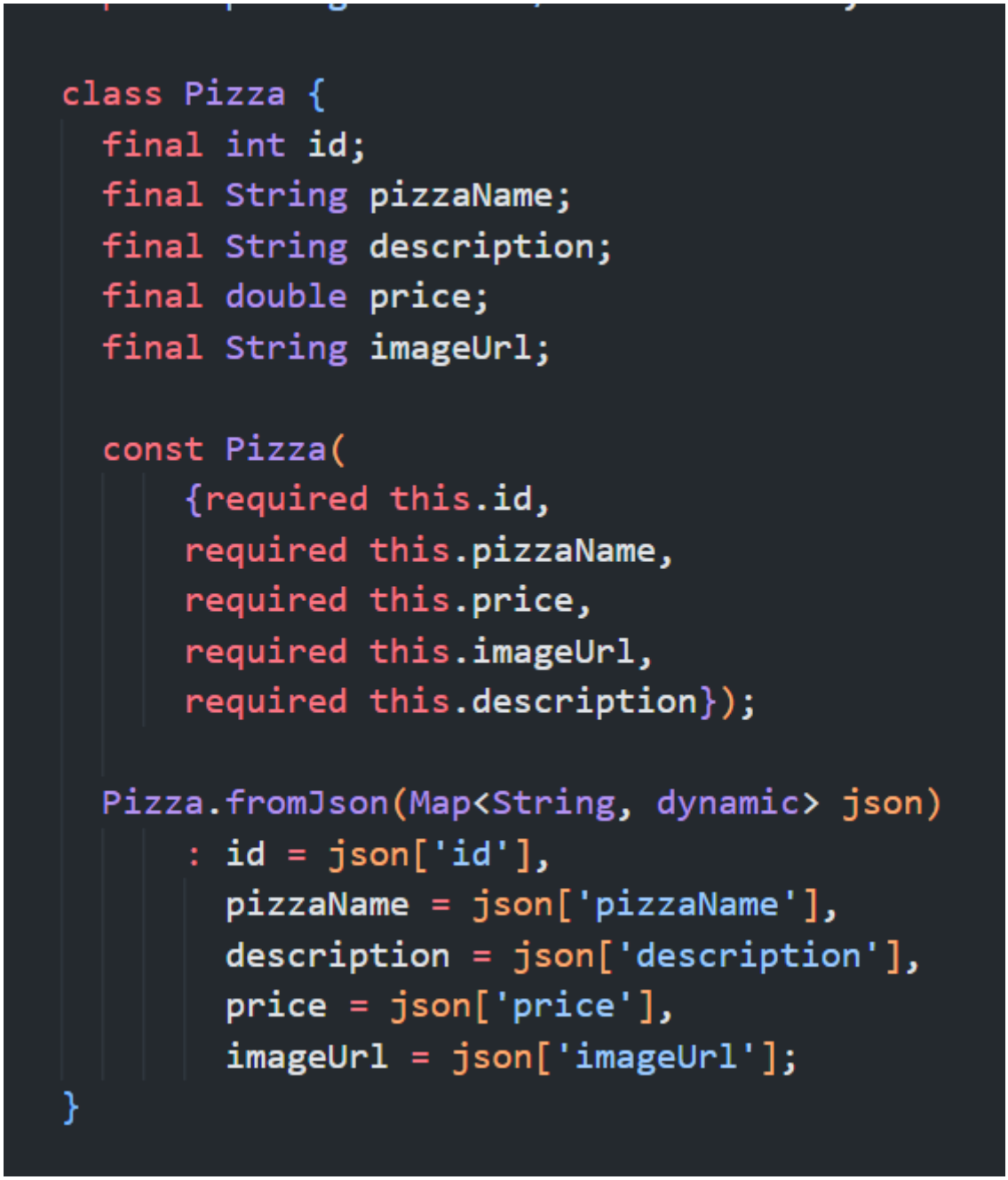
13. Di dalam kelas Pizza, tentukan konstruktor bernama fromJson, yang akan mengambil sebuah Map sebagai parameter dan mengubah Map menjadi sebuah instance dari Pizza:

14. Refaktor metode readJsonFile() pada kelas _MyHomePageState. Langkah pertama adalah mengubah String menjadi Map dengan memanggil metode jsonDecode. Pada method readJsonFile, tambahkan kode yang di cetak tebal berikut ini:



15. Pastikan editor Anda secara otomatis menambahkan pernyataan impor untuk pustaka "dart:convert" di bagian atas file main.dart; jika tidak, tambahkan saja secara manual. Tambahkan juga pernyataan impor

untuk kelas pizza:



```dart
class Pizza {
  final int id;
  final String pizzaName;
  final String description;
  final double price;
  final String imageUrl;

  const Pizza(
      {required this.id,
      required this.pizzaName,
      required this.price,
      required this.imageUrl,
      required this.description});

  Pizza.fromJson(Map<String, dynamic> json)
      : id = json['id'],
        pizzaName = json['pizzaName'],
        description = json['description'],
        price = json['price'],
        imageUrl = json['imageUrl'];
}
```

> Saya jadikan di satu file di main

16. Langkah terakhir adalah mengonversi string JSON kita menjadi List of native Dart objects. Kita dapat melakukan ini dengan mengulang pizzaMapList dan mengubahnya menjadi objek Pizza. Di dalam metode readJsonFile, di bawah metode jsonDecode, tambahkan kode berikut:

```dart
List<Pizza> myPizzas = pizzaMapList.map((pizza)=> Pizza.fromJson(pizza)).toList();
```

17. Hapus atau beri komentar pada metode setState yang mengatur String pizzaString dan kembalikan daftar objek Pizza sebagai gantinya:

```
Future readJsonFile() async {
  String myString = await DefaultAssetBundle.
      .loadString('images/pizzaList.json');
  List pizzaMapList = jsonDecode(myString);
  List<Pizza> myPizzas = pizzaMapList.map((pi

  return myPizzas;
}
```

18. Ubah signature metode sehingga Anda dapat menampilkan nilai balik secara eksplisit:

```
Future<List<Pizza>> readJsonFile() async {
  String myString = await DefaultAssetBundle.of(
      .loadString('images/pizzaList.json');
  List pizzaMapList = jsonDecode(myString);
  List<Pizza> myPizzas = pizzaMapList.map((pizza

  return myPizzas;
}
```

19. Sekarang kita memiliki objek List of Pizza. Daripada hanya menampilkan sebuah Teks kepada pengguna, kita dapat menampilkan sebuah ListView yang berisi sekumpulan widget ListTile. Di bagian atas kelas _MyHomePageState, buat `List<Pizza>` bernama myPizzas:

```
List<Pizza> myPizzas = [];
```

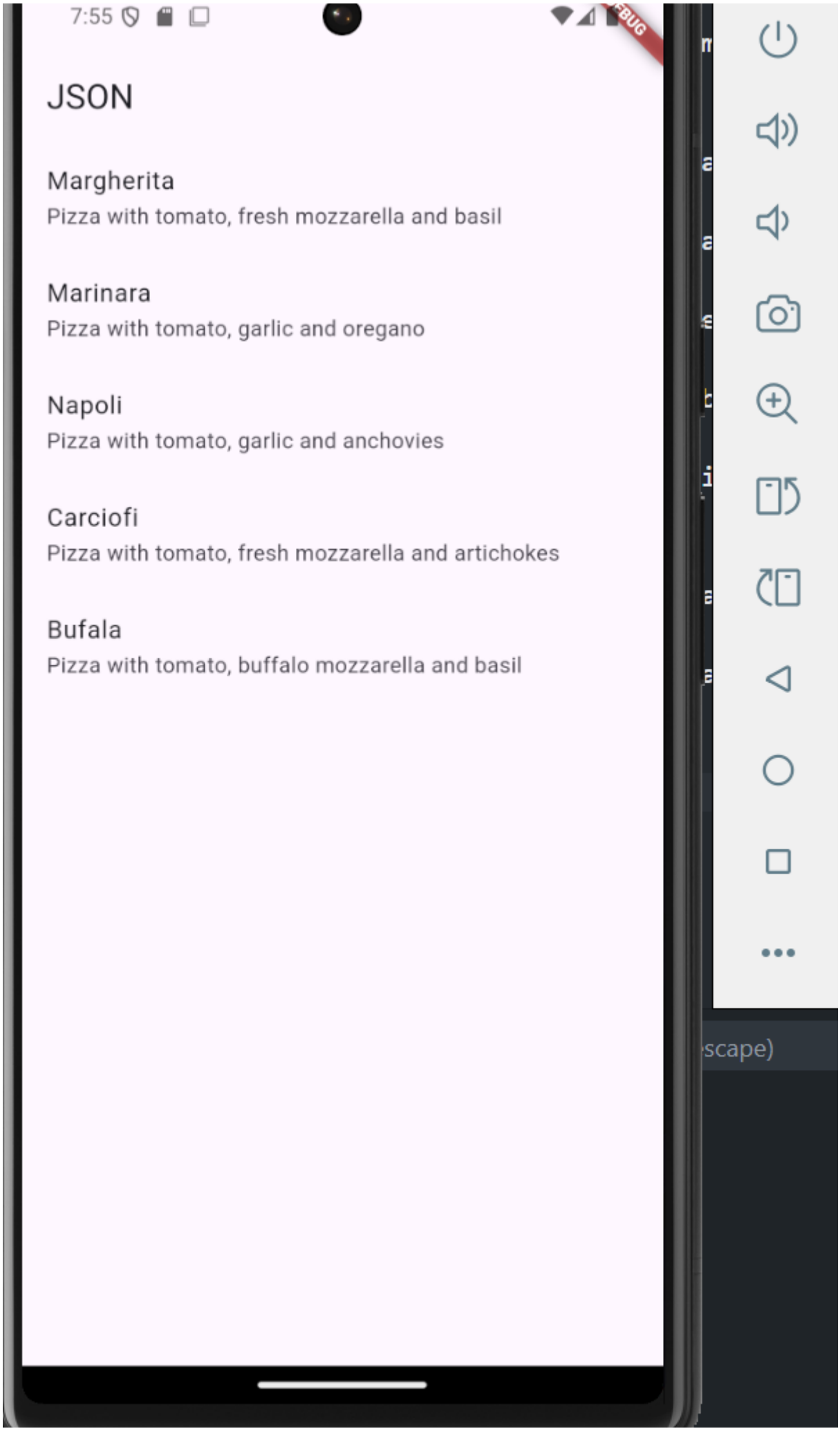
20. Dalam metode initState, pastikan Anda mengatur myPizzas dengan hasil panggilan ke readJsonFile:

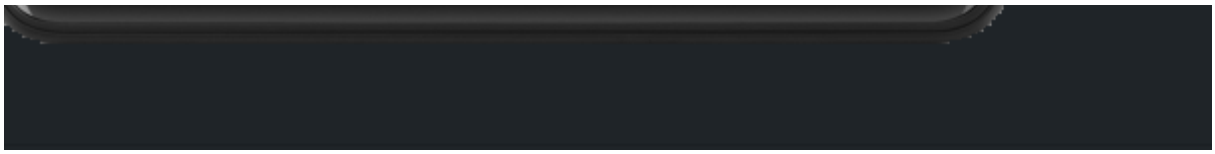
```
@override
void initState() {
  super.initState();
  readJsonFile().then((value){
    setState(() {
      myPizzas = value;
    });
  });
}
```

21. Tambahkan kode berikut ini di dalam Scaffold, di dalam metode build():

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('JSON'),
    ), // AppBar
    body: Center(
      child: ListView.builder(
        itemCount: myPizzas.length,
        itemBuilder: (context,index){
          Pizza myPizza = myPizzas[index];
          return ListTile(
            title: Text(myPizza.pizzaName),
            subtitle: Text(myPizza.description),
          ); // ListTile
          }
      ), // ListView.builder
    ), // Center
  ); // Scaffold
```

22. Jalankan aplikasi. Antarmuka pengguna sekarang seharusnya jauh lebih ramah dan terlihat seperti yang ditunjukkan pada


7:55
JSON
Margherita
Pizza with tomato, fresh mozzarella and basil
Marinara
Pizza with tomato, garlic and oregano
Napoli
Pizza with tomato, garlic and anchovies
Carciofi
Pizza with tomato, fresh mozzarella and artichokes
Bufala
Pizza with tomato, buffalo mozzarella and basil



Praktikum 2

1. Tambahkan metode baru ke kelas Pizza, di file pizza.dart, yang disebut toJson. Ini akan mengembalikan sebuah `Map<String, dynamic>` dari objek:

```
Map<String,dynamic> toJson(){
  return {
   'id' : id,
   'pizzaName' : pizzaName ,
   'description' : price ,
   'price' : imageUrl ,
   'imageUrl' : description
  };
 }
```

2. Setelah Anda memiliki sebuah Map, Anda dapat menserialisasikannya kembali ke dalam string JSON. Tambahkan metode baru di di bagian bawah kelas _MyHomePageState, di dalam file main.dart, yang disebut convertToJSON:

```
String convertToJson(List<Pizza> pizzas) {
  return jsonEncode(pizzas.map((pizza) => jsonEncode(pizza)).toList());
 }
```

3. Metode ini mengubah objek List of Pizza kembali menjadi string Json dengan memanggil metode jsonEncode lagi di pustaka dart_convert.
4. Terakhir, mari panggil metode tersebut dan cetak string JSON di Debug Console. Tambahkan kode berikut ke metode readJsonFile, tepat sebelum mengembalikan List myPizzas:

```
String json = convertToJson(myPizzas);
print(json);    Don't invoke 'print' in p
```

5. Jalankan aplikasi. Anda akan melihat string JSON dicetak, seperti yang ditunjukkan pada gambar berikut:

```
Restarted application in 2,502ms.
I/flutter ( 3459): ["{\"id\":1,\"pizzaName\":\"Margherita\",\"description\":8.75,\"price\":\"images/margherita.png\",\"imageUrl\":\"Pizz
sil\"}","{\"id\":2,\"pizzaName\":\"Marinara\",\"description\":7.5,\"price\":\"images/marinara.png\",\"imageUrl\":\"Pizza with tomato, ga
aName\":\"Napoli\",\"description\":9.5,\"price\":\"images/marinara.png\",\"imageUrl\":\"Pizza with tomato, garlic and anchovies\"}","{\"
scription\":8.8,\"price\":\"images/marinara.png\",\"imageUrl\":\"Pizza with tomato, fresh mozzarella and artichokes\"}","{\"id\":5,\"piz
2.5,\"price\":\"images/marinara.png\",\"imageUrl\":\"Pizza with tomato, buffalo mozzarella and basil\"}"]
D/EGL_emulation( 3459): app_time_stats: avg=571054.69ms min=571054.69ms max=571054.69ms count=1
```

Praktikum 3

1. Gunakan project pada pertemuan 11 bernama books. Pertama, tambahkan ketergantungan pada shared_preferences. Dari Terminal Anda, ketikkan perintah berikut

```
PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 12\books> flutter pub add shared_preferences
Resolving dependencies...
Downloading packages... (1.1s)
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.18.0 (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
+ ffi 2.1.3
+ file 7.0.1
  flutter_lints 4.0.0 (5.0.0 available)
  geolocator 13.0.1 (13.0.2 available)
```

2. Untuk memperbarui dependensi dalam proyek Anda, jalankan perintah flutter pub get dari jendela Terminal.

```
  lints 4.0.0 (5.1.0 available)
PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 12\books> flutter pub get
Resolving dependencies...
Downloading packages...
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.3.1 available)
  clock 1.1.1 (1.1.2 available)
```

3. Di bagian atas file main.dart, impor shared_preferences:

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

4. Di bagian atas kelas _MyHomePageState, buat variabel status integer baru bernama appCounter:

```
int appCounter = 0;
```

5. Dalam kelas _MyHomePageState, buat metode asinkron baru yang disebut readAndWritePreferences():

```
60
61    Future readAndWritePreference() async{
62
63    }
```

6. Di dalam metode readAndWritePreference, buatlah sebuah instance dari SharedPreferences:

```
SharedPreferences prefs = await SharedPreferences.getInstance();
```

7. Setelah membuat instance preferensi, kita membuat kode yang mencoba baca nilai kunci appCounter. Jika nilainya nol, setel ke 0; lalu naikkan nilainya:

```
appCounter = prefs.getInt('appCounter') ?? 0 ;
appCounter++;
```

8. Setelah itu, atur nilai kunci appCounter di preferensi ke nilai baru:

```
await prefs.setInt('appCounter', appCounter);
```

9. Memperbarui nilai status appCounter

```
setState(() {
  appCounter = appCounter;
});
```

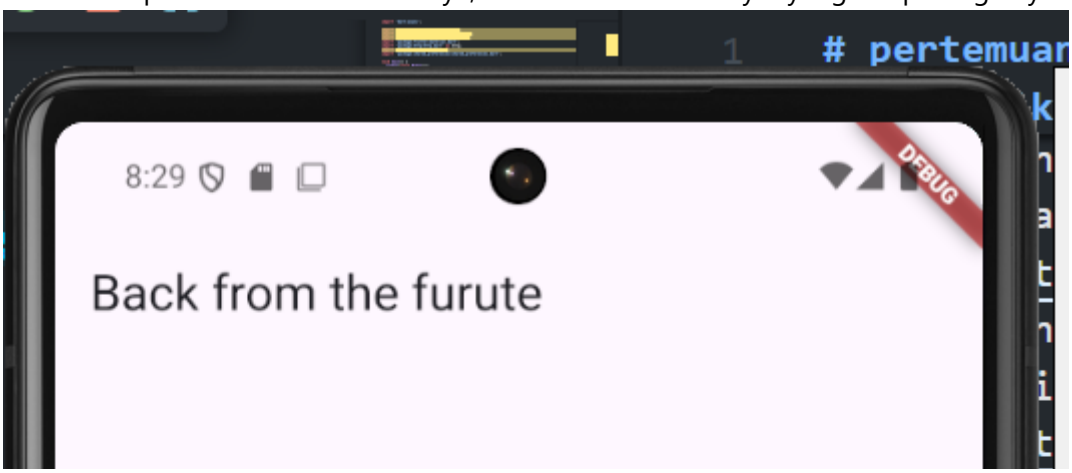
10. Pada metode initState di kelas _MyHomePageState, panggil metode readAndWritePreference() dengan kode yang dicetak tebal:

```
@override
void initState() {
 readAndWritePreference();
  super.initState();
}
```

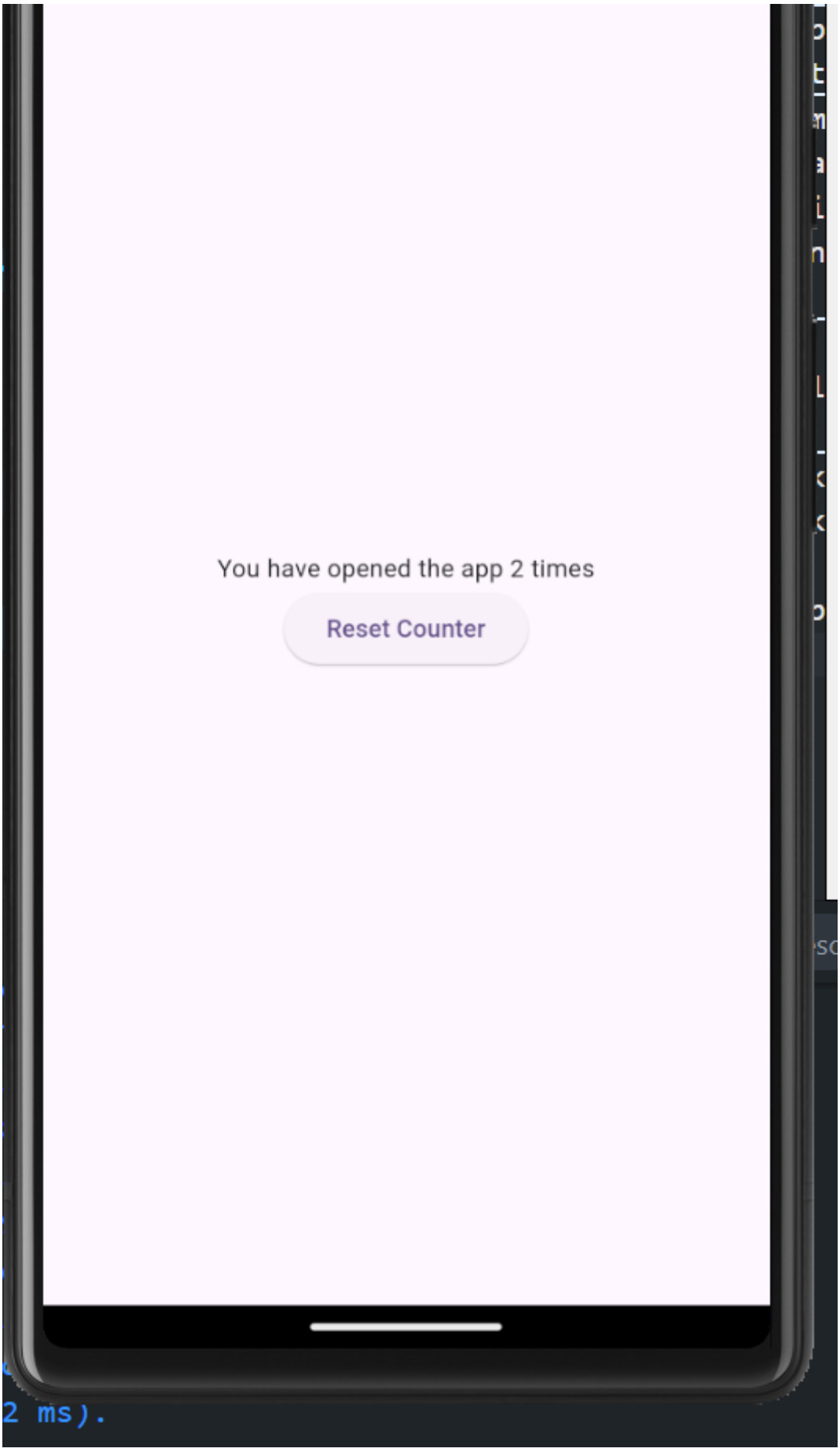
11. Dalam metode build, tambahkan kode berikut ini di dalam widget Container:

```
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("Back from the furute"),
        ), // AppBar
        body: Center(
            child: Center(
          child: Column(
            children: [
              Text('You have opened the app $appCounter times'),
              ElevatedButton(onPressed: () {}, child: Text('Reset Counter'))
            ],
          ), // Column
        ))); // Center // Center // Scaffold
  }
}
```

12. Jalankan aplikasi. Saat pertama kali membukanya, Anda akan melihat layar yang mirip dengan yang



berikut ini:



13. Tambahkan metode baru ke kelas _MyHomePageState yang disebut deletePreference(), yang akan menghapus nilai yang disimpan:

```
71    Future deletePreference() async{
72      SharedPreferences prefs = await SharedPreferences.getInstance();
73      await prefs.clear();
74      setState(() {
75        appCounter=0;
76      });
77    }
```

14. Dari properti onPressed dari widget ElevatedButton di metode build(), memanggil metode deletePreference(), dengan kode di cetak tebal:

```
children: [
  Text('You have opened the app $appCounter times'),
  ElevatedButton(
      onPressed: () {
        deletePreference();
      },
```

15. jalankan aplikasi lagi

Praktikum 4

1. Buat project baru

```
PS D:\Kuliahe wong jenius\Semester 5\Mobile\Pertemuan 14> flutter create path_provider
Creating project path_provider...
Resolving dependencies in `path_provider`... (1.5s)
Downloading packages...
Got dependencies in `path_provider`.
Wrote 129 files.

All done!
You can find general documentation for Flutter at: https://docs.flutter.dev/
Detailed API documentation is available at: https://api.flutter.dev/
If you prefer video documentation, consider: https://www.youtube.com/c/flutterdev

In order to run your application, type:

  $ cd path_provider
  $ flutter run

Your application code is in path_provider\lib\main.dart.
```

2. menambahkan dependency yang relevan ke file pubspec.yaml. Tambahkan path_provider dengan mengetikkan perintah ini dari Terminal Anda:

```
# versions available, run `flutter pub outdated`.
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^1.0.8
  path_provider: ^2.1.5
```

3. Di bagian atas file main.dart, tambahkan impor path_provider:

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

4. Di bagian atas kelas _MyHomePageState, tambahkan variabel State yang akan kita gunakan untuk memperbarui antarmuka pengguna:

```
class _MyHomePageState extends State<MyHomePage> {
  String documentPath = '';
  String tempPath = '';
```

5. Masih dalam kelas _MyHomePageState, tambahkan metode untuk mengambil direktori temporary dan dokumen:

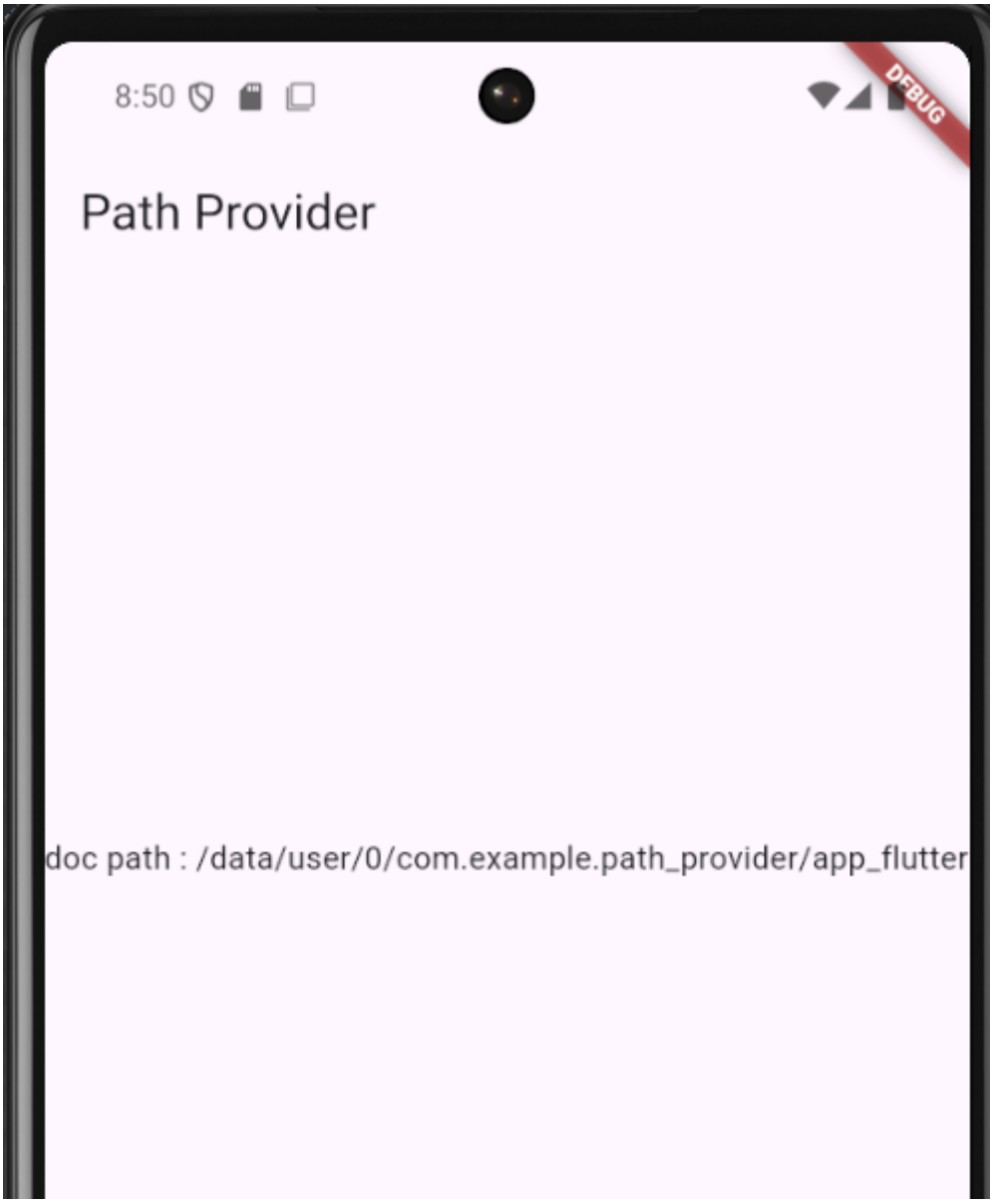
```
  Future getPaths() async {
    final docDir = await getApplicationDocumentsDirectory();
    final tempDir = await getTemporaryDirectory();

    setState(() {
      documentPath = docDir.path;
      tempPath = tempDir.path;
    });
  }
```

6. Pada metode initState dari kelas _MyHomePageState, panggil metode getPaths:

```
  @override
  void initState() {
    getPaths();
    super.initState();
  }
```

7. Pada metode build _MyHomePageState, buat UI dengan dua widget Teks yang menunjukkan path yang diambil:

```dart
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Path Provider'),
      ), // AppBar
      body: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          Text('doc path : $documentPath'),
          Text('temp path : $tempPath')
        ],
      ), // Column
    ); // Scaffold
  }
}
```



8. Jalankan aplikasi 

Praktikum 5

1. Di bagian atas berkas main.dart, impor pustaka dart:io:

```
import 'dart:io';
```

2. Di bagian atas kelas _MyHomePageState, di file main.dart, buat dua variabel State baru untuk file dan isinya:

```
late File myFile;
String fileText ='';
```

3. Masih dalam kelas MyHomePageState, buat metode baru bernama writeFile dan gunakan kelas File dari pustaka dart:io untuk membuat file baru:

```
Future<bool> writeFile() async{
  try{
    await myFile.writeAsString('Margherita, Capricciosa, Napoli');
    return true;

  }catch(e){
    return false;
  }
}
```

4. Dalam metode initState, setelah memanggil metode getPaths, dalam metode then, buat sebuah file dan panggil metode writeFile:

```
@override
void initState() {
  getPaths().then((_) {
    myFile = File('$documentPath/pizzas.txt');
    writeFile();
  });
  super.initState();
}
```

5. Buat metode untuk membaca file:

```
82
83      Future<bool> readFile()async{
84        try{
85          String fileContent = await myFile.readAsString();
86          setState(() {
87            fileText = fileContent;
88          });
89
90          return true;
91        }catch(e){
92            return false;
93        }
94      }
95
```

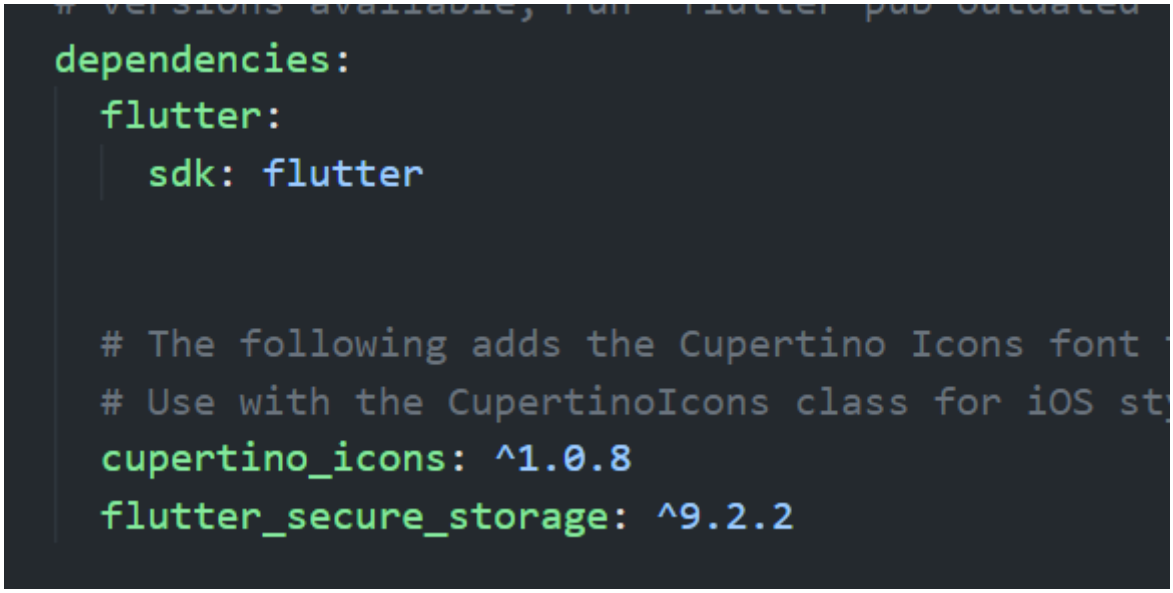
6. Dalam metode build, di widget Column, perbarui antarmuka pengguna dengan ElevatedButton. Ketika pengguna menekan tombol, tombol akan mencoba membaca konten file dan menampilkannya di layar,

cek kode cetak tebal:

7. Jalankan aplikasi dan tekan tombol Baca File. Di bawah tombol tersebut, Anda akan melihat teks Margherita, Capricciosa, Napoli, seperti yang ditunjukkan pada tangkapan layar berikut:

Praktikum 6

1. Install dependency



2. Di file main.dart, salin kode berikut:

```
54
55    @override
56    Widget build(BuildContext context) {
57      return Scaffold(
58        appBar: AppBar(
59          title: const Text('Path Provider'),
60        ), // AppBar
61        body: SingleChildScrollView(
62          child: Padding(
63            padding: const EdgeInsets.all(16.0),
64            child: Column(
65              children: [
66                TextField(
67                  controller: pwdController,
68                ), // TextField
69                ElevatedButton(onPressed: () {}, child: const Text('Save value')),
70                ElevatedButton(onPressed: () {}, child: const Text('Read Value')),
71              ],
72            ), // Column
73          ), // Padding
74        ), // SingleChildScrollView
75      ); // Scaffold
76    }
77  }
78
```

3. Di bagian atas file main.dart, tambahkan impor yang diperlukan:

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

4. Di bagian atas kelas _myHomePageState, buat penyimpanan yang aman:

```
46
47  class _MyHomePageState extends State<MyHomePage> {
48    final storage = const FlutterSecureStorage();
49    final key = 'myPass';
50    final pwdController = TextEditingController();
51    String myPass = '';
52
```

5. Di kelas _myHomePageState, tambahkan metode untuk menulis data ke penyimpanan aman:

```
52
53    Future writeToSecureStorage() async{
54      await storage.write(key: key, value: pwdController.text);
55    }
56
```

6. Pada metode build() dari kelas _myHomePageState, tambahkan kode yang akan menulis ke penyimpanan ketika pengguna menekan tombol Save Value, cek kode cetak tebal:

```
children: [
  TextField(
    controller: pwdController,
  ), // TextField
  ElevatedButton(onPressed: () {
    writeToSecureStorage();
  }, child: const Text('Save value')), // ElevatedButton
```

7. Di kelas _myHomePageState, tambahkan metode untuk membaca data dari penyimpanan aman:

```
56
57      Future<String> readFromSecureStorage() async{
58        String secret = await storage.read(key: key) ?? '';
59        return secret;
60      }
```

8. Pada metode build() dari kelas _myHomePageState, tambahkan kode untuk membaca dari penyimpanan ketika pengguna menekan tombol Read Value dan memperbarui variabel myPass State:

```
        child: const Text('Save value')), // ElevatedButton
    ElevatedButton(
        onPressed: () {
          readFromSecureStorage().then((value) {
            setState(() {
              myPass = value;
            });
          });
        },
        child: const Text('Read Value')), // ElevatedButton
  ],
), // Column
```

9. Jalankan aplikasi dan tulis beberapa teks pilihan Anda di bidang teks. Kemudian, tekan tombol Save Value. Setelah itu, tekan tombol Read Value. Anda akan melihat teks yang Anda ketik di kolom teks, seperti yang ditunjukkan pada tangkapan layar berikut: