

Laporan Praktikum

Mobile Programming

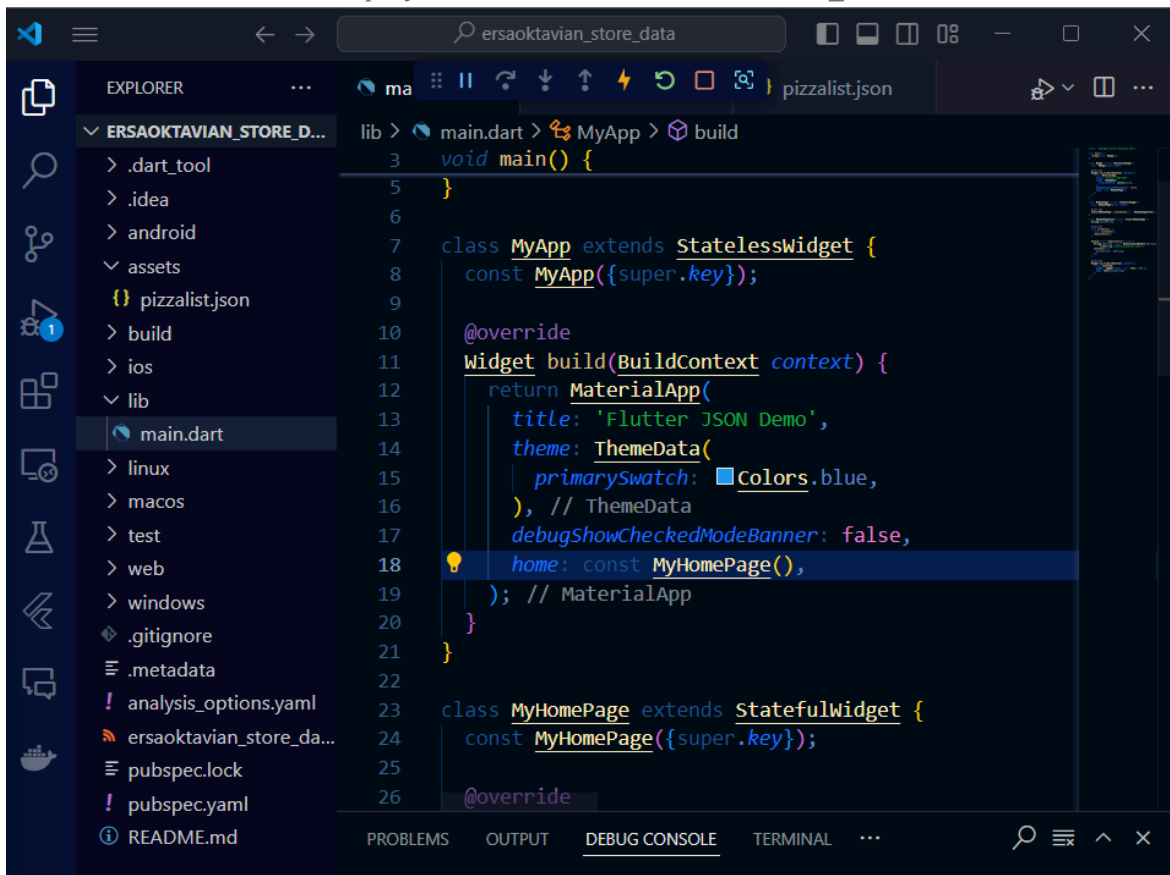
NAMA : ERSO OKTAVIAN RAMADAN

NIM : 2241720208

KELAS : TI - 2B / 09

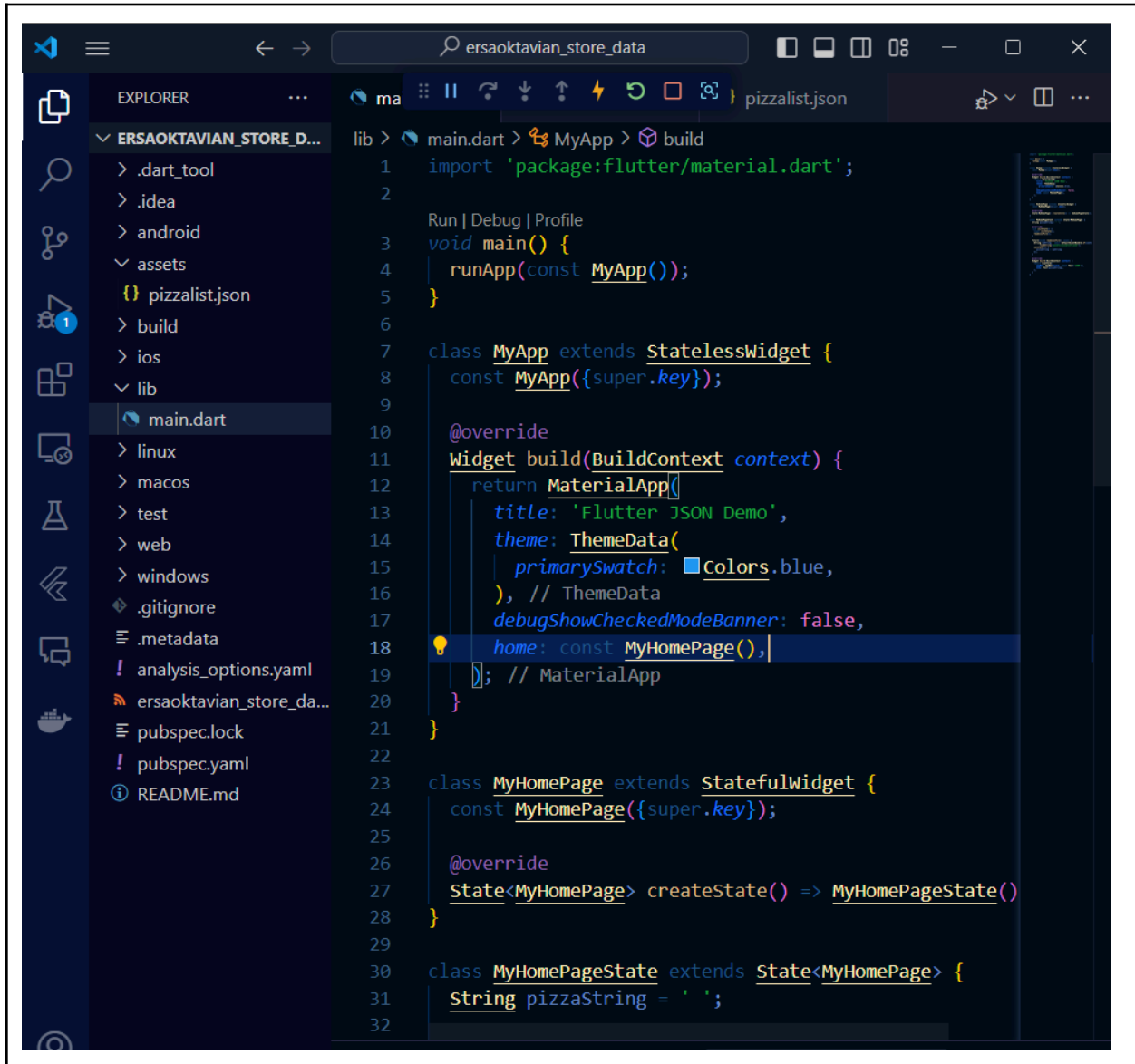
A. Praktikum 1: Converting Dart models into JSON

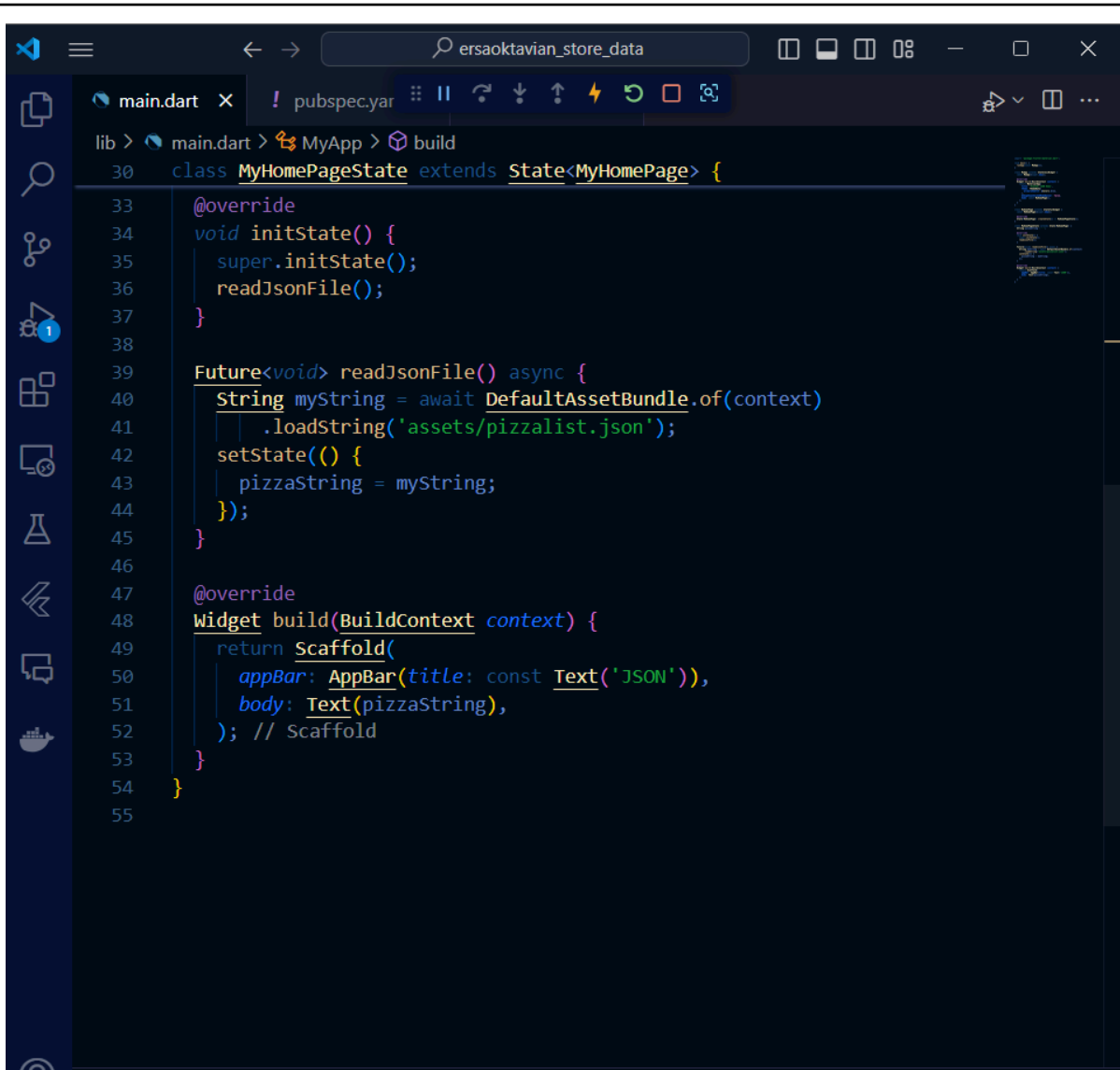
1. Di editor favorit Anda, buat proyek Flutter baru dan beri nama store_data



```
lib > main.dart > MyApp > build
3 void main() {
5 }
6
7 class MyApp extends StatelessWidget {
8   const MyApp({super.key});
9
10  @override
11  Widget build(BuildContext context) {
12    return MaterialApp(
13      title: 'Flutter JSON Demo',
14      theme: ThemeData(
15        primarySwatch: Colors.blue,
16      ), // ThemeData
17      debugShowCheckedModeBanner: false,
18      home: const MyHomePage(),
19    ); // MaterialApp
20  }
21 }
22
23 class MyHomePage extends StatefulWidget {
24   const MyHomePage({super.key});
25
26   @override
```

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

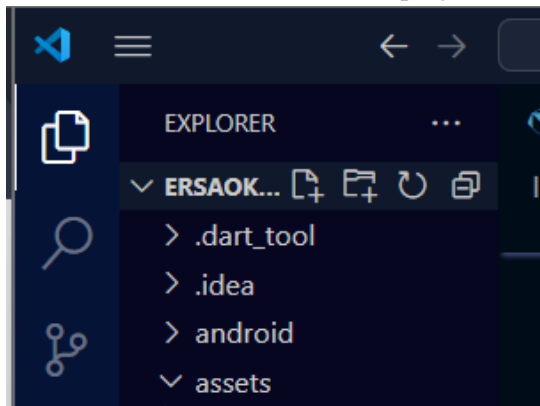




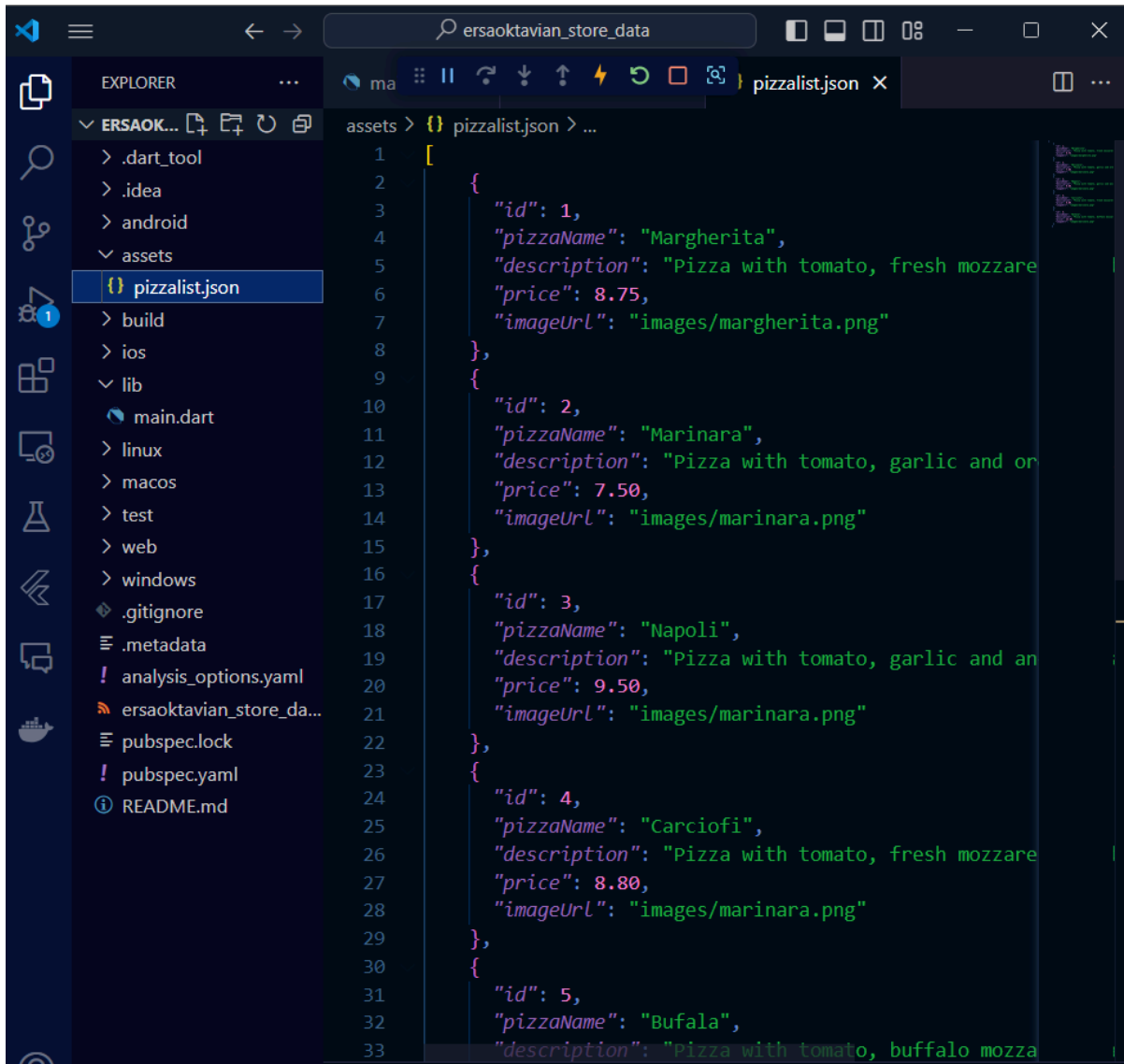
The screenshot shows an IDE window with a search bar at the top containing 'ersaoktavian_store_data'. Below the search bar, there are tabs for 'main.dart' and 'pubspec.yaml'. The main.dart tab is active, displaying the following Dart code:

```
lib > main.dart > MyApp > build
30 class MyHomePageState extends State<MyHomePage> {
31
32
33   @override
34   void initState() {
35     super.initState();
36     readJsonFile();
37   }
38
39   Future<void> readJsonFile() async {
40     String myString = await DefaultAssetBundle.of(context)
41       .loadString('assets/pizzalist.json');
42     setState(() {
43       pizzaString = myString;
44     });
45   }
46
47   @override
48   Widget build(BuildContext context) {
49     return Scaffold(
50       appBar: AppBar(title: const Text('JSON')),
51       body: Text(pizzaString),
52     ); // Scaffold
53   }
54 }
55
```

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

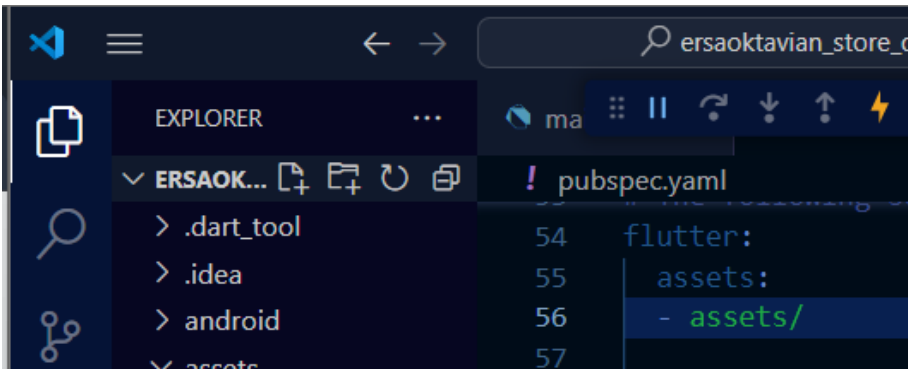


4. Di dalam folder aset, buat file baru bernama pizzalist.json dan salin konten yang tersedia di tautan <https://gist.github.com/simoales/a33c1c2abe78b48a75ccfd5fa0de0620>. File ini berisi daftar objek JSON.

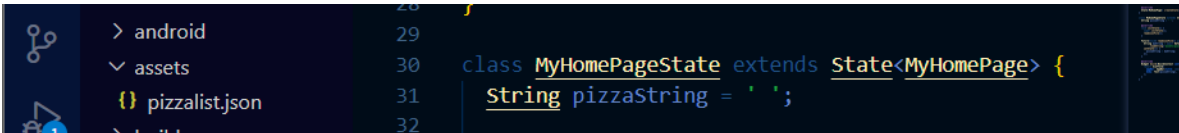


```
1  [
2
3      {
4          "id": 1,
5          "pizzaName": "Margherita",
6          "description": "Pizza with tomato, fresh mozzarella",
7          "price": 8.75,
8          "imageUrl": "images/margherita.png"
9      },
10
11     {
12         "id": 2,
13         "pizzaName": "Marinara",
14         "description": "Pizza with tomato, garlic and onion",
15         "price": 7.50,
16         "imageUrl": "images/marinara.png"
17     },
18
19     {
20         "id": 3,
21         "pizzaName": "Napoli",
22         "description": "Pizza with tomato, garlic and anchovies",
23         "price": 9.50,
24         "imageUrl": "images/marinara.png"
25     },
26
27     {
28         "id": 4,
29         "pizzaName": "Carciofi",
30         "description": "Pizza with tomato, fresh mozzarella and artichokes",
31         "price": 8.80,
32         "imageUrl": "images/marinara.png"
33     },
34
35     {
36         "id": 5,
37         "pizzaName": "Bufala",
38         "description": "Pizza with tomato, buffalo mozzarella and anchovies",
39         "price": 9.50,
40         "imageUrl": "images/marinara.png"
41     }
42 ]
```

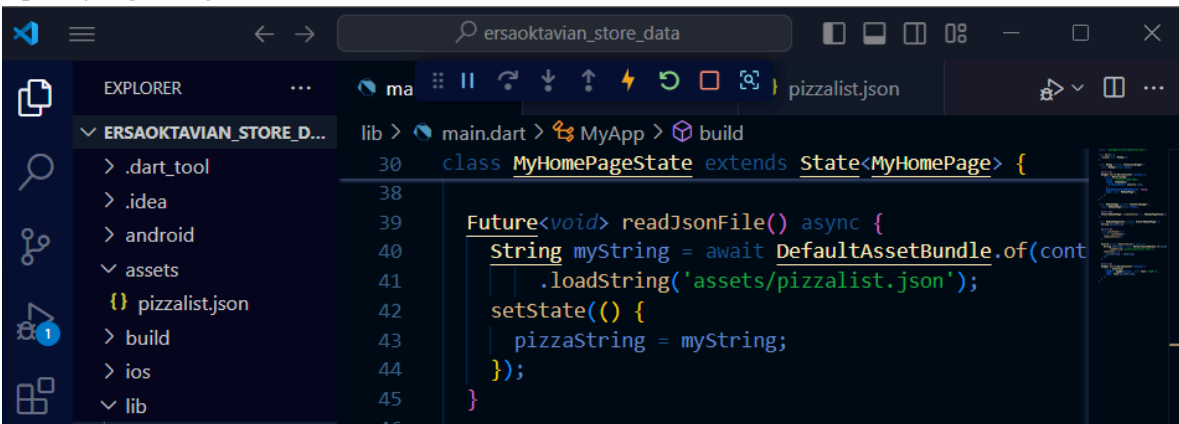
5. Di file pubspec.yaml, tambahkan referensi ke folder aset baru, seperti yang ditunjukkan di sini:



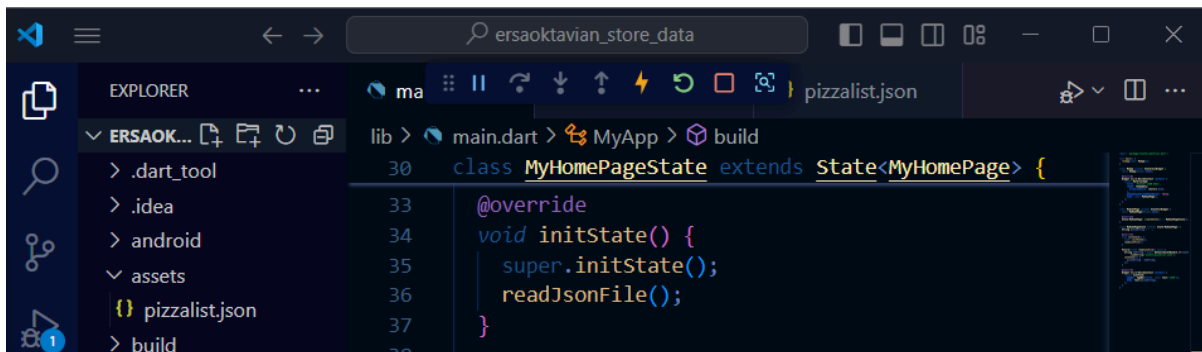
6. Pada kelas _MyHomePageState, di main.dart, tambahkan sebuah variabel state bernama pizzaString:



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:

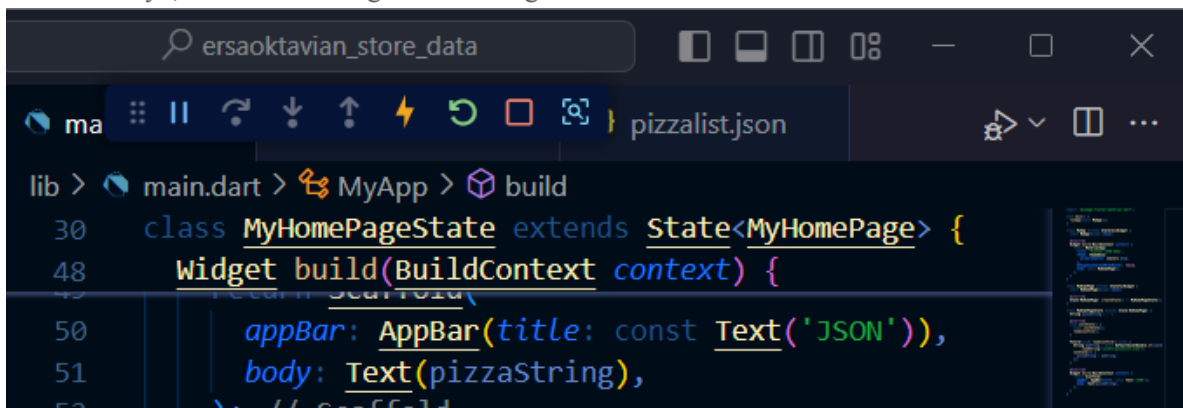


8. Pada kelas `_MyHomePageState`, timpa metode `initState` dan, di dalamnya, panggil metode `readJsonFile`:



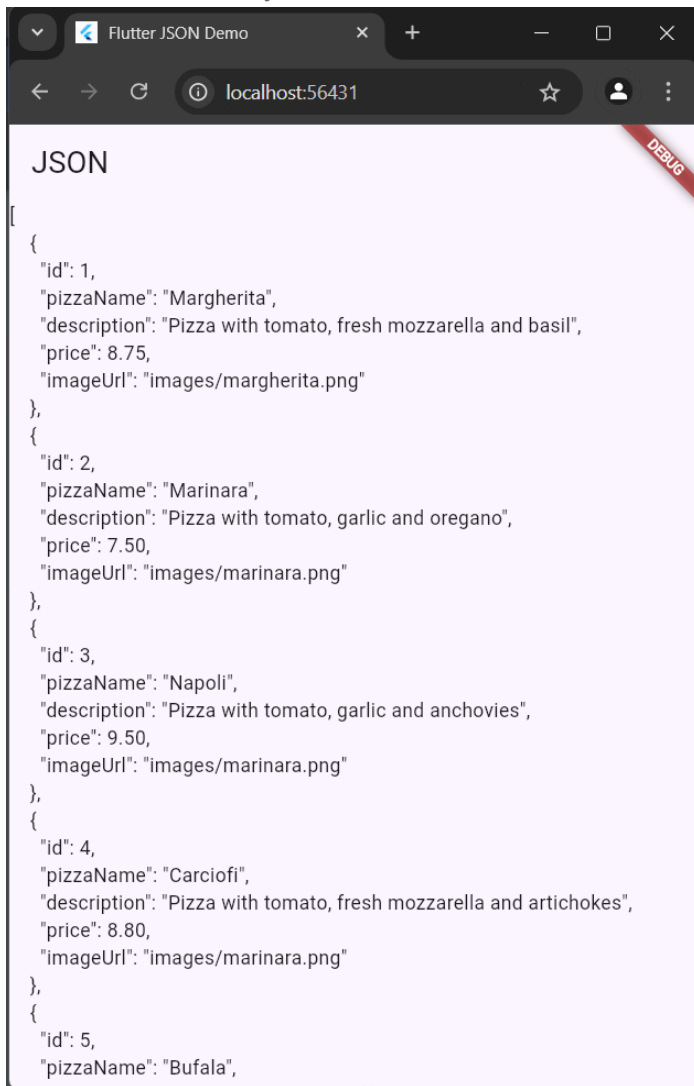
```
lib > main.dart > MyApp > build
30 class MyHomePageState extends State<MyHomePage> {
33   @override
34   void initState() {
35     super.initState();
36     readJsonFile();
37   }
38 }
```

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

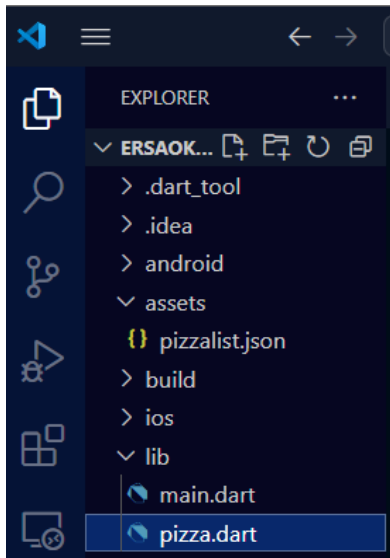


```
lib > main.dart > MyApp > build
30 class MyHomePageState extends State<MyHomePage> {
48   Widget build(BuildContext context) {
49     // readJsonFile()
50     appBar: AppBar(title: const Text('JSON')),
51     body: Text(pizzaString),
52   } // Scaffold
}
```

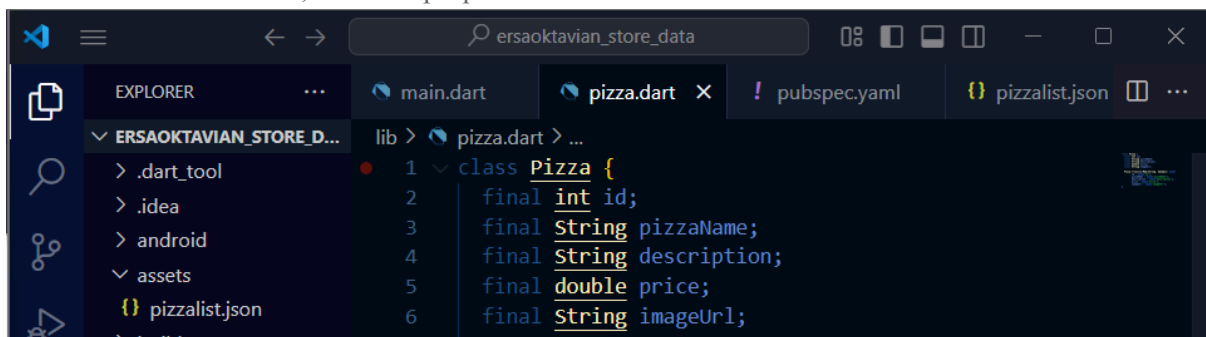
10. Mari kita jalankan aplikasinya. Jika semuanya berjalan seperti yang diharapkan, Anda akan melihat konten file JSON di layar



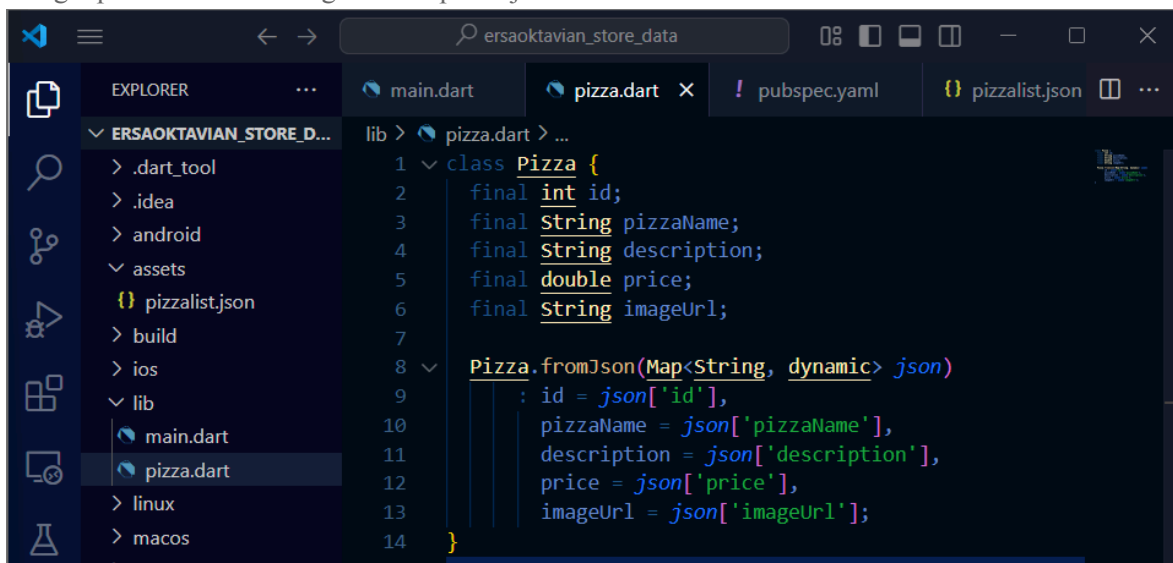
11. Kita ingin mengubah String ini menjadi sebuah List of Objects. Kita akan mulai dengan membuat kelas baru. Dalam folder lib aplikasi kita, buat file baru bernama pizza.dart.



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



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 berikut ini:

```
Pizza.fromJson(Map<String, dynamic> json)
```

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`:

```
lib > main.dart > MyHomePageState > build
1 import 'dart:convert';
2 import 'package:flutter/material.dart';
3 import './pizza.dart';
```

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:

```
Future<List<Pizza>> readJsonFile() async {
  String myString = await DefaultAssetBundle.of(context)
    .loadString('assets/pizzalist.json');
  List pizzaMapList = jsonDecode(myString); // Decode JSON
  List<Pizza> pizzas = [];
  for (var pizza in pizzaMapList) {
    pizzas.add(Pizza.fromJson(pizza)); // Konversi Map ke Pizza
  }
  return pizzas;
}
```

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

```
main.dart
52 }
53 return pizzas;
54 }
```

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

```
Future<List<Pizza>> readJsonFile() async {
```

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 bernama `myPizzas`:

```
List<Pizza> myPizzas = [];
List<Pizza> pizzas = [];
for (var pizza in pizzaMapList) {
```

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



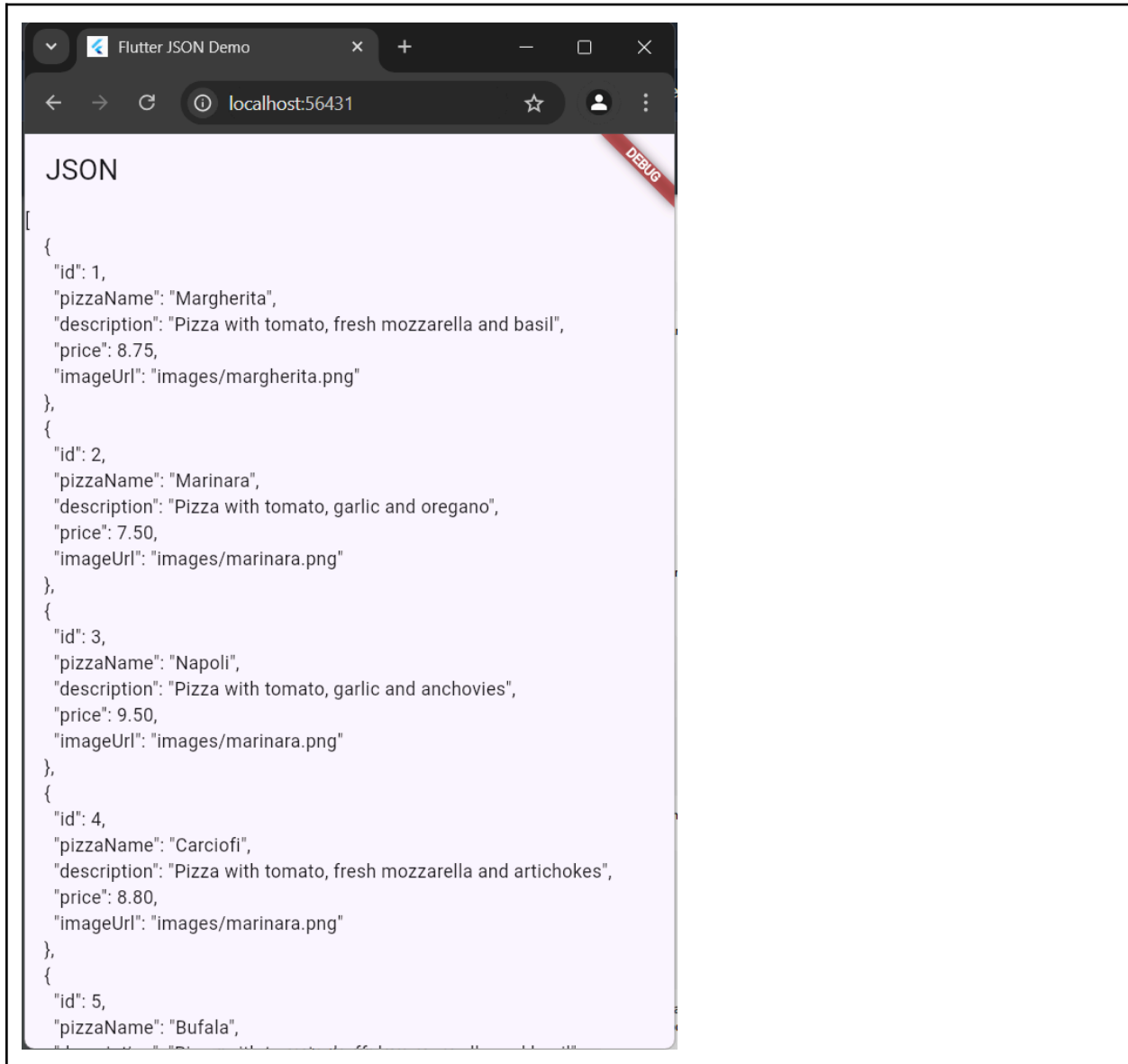
```
35 @override
36 void initState() {
37   super.initState();
38   readJsonFile().then((value) {
39     setState(() {
40       myPizzas = value;
41     });
42   });
43 }
44
```

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



```
57 Widget build(BuildContext context) {
58   // AppBar
59   body: ListView.builder(
60     itemCount: myPizzas.length,
61     itemBuilder: (context, index) {
62       return ListTile(
63         title: Text(myPizzas[index].pizzaName),
64         subtitle: Text(
65           '${myPizzas[index].description} - €${myP
66         }, // Text
67       ); // ListTile
68     }, // ListView.builder
69   ); // Scaffold
70 }
71
```

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



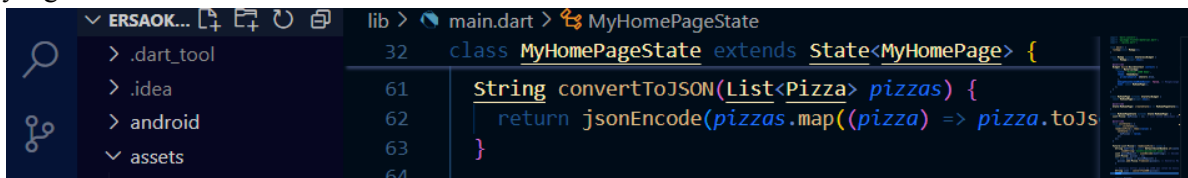
Praktikum 2: Reading the JSON file

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



```
1 class Pizza {
25   Map<String, dynamic> toJson() {
27       'id': id,
28       'pizzaName': pizzaName,
29       'description': description,
30       'price': price,
31       'imageUrl': imageUrl,
32   };
33 }
34
35
```

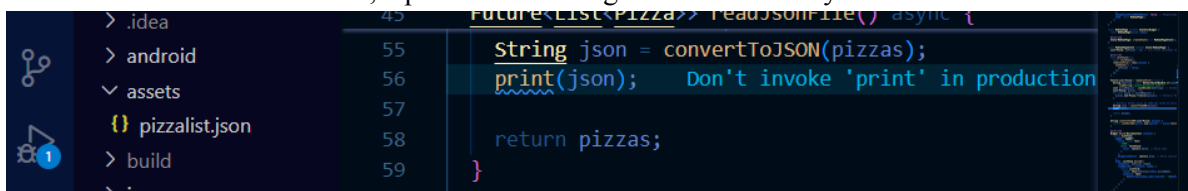
2. Setelah Anda memiliki sebuah Map, Anda dapat menserialisasikannya kembali ke dalam string JSON. Tambahkan metode baru di bagian bawah kelas _MyHomePageState, di dalam file main.dart, yang disebut convertToJson:



```
32 class _MyHomePageState extends State<MyHomePage> {
61   String convertToJson(List<Pizza> pizzas) {
62       return jsonEncode(pizzas.map((pizza) => pizza.toJson()));
63   }
64 }
```

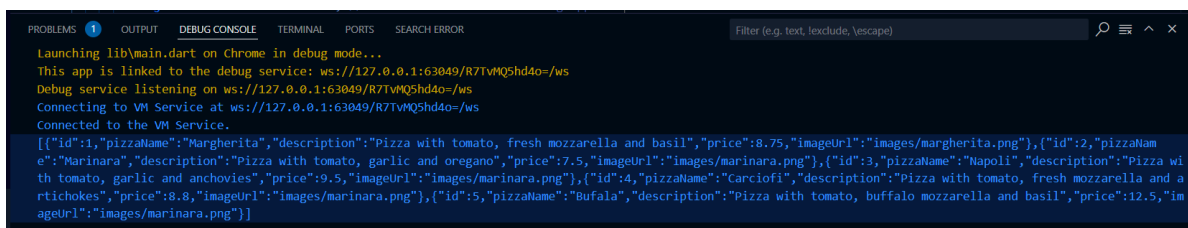
3. Metode ini mengubah objek List of Pizza kembali menjadi string Json dengan memanggil metode jsonEncode lagi di pustaka dart_convert.

4. Terakhir, mari panggil metode tersebut dan cetak string JSON di Debug Console. Tambahkan kode berikut ke metode readJsonFile, tepat sebelum mengembalikan List myPizzas:



```
45 Future<List<Pizza>> readJsonFile() async {
55   String json = convertToJson(pizzas);
56   print(json); // Don't invoke 'print' in production
57
58   return pizzas;
59 }
```

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



```
Launching lib/main.dart on Chrome in debug mode...
This app is linked to the debug service: ws://127.0.0.1:63049/R71vMQ5hd4o=/ws
Debug service listening on ws://127.0.0.1:63049/R71vMQ5hd4o=/ws
Connecting to VM Service at ws://127.0.0.1:63049/R71vMQ5hd4o=/ws
Connected to the VM Service.
[{"id":1,"pizzaName":"Margherita","description":"Pizza with tomato, fresh mozzarella and basil","price":8.75,"imageUrl":"images/margherita.png"}, {"id":2,"pizzaName":"Marinara","description":"Pizza with tomato, garlic and oregano","price":7.5,"imageUrl":"images/marinara.png"}, {"id":3,"pizzaName":"Napoli","description":"Pizza with tomato, garlic and anchovies","price":9.5,"imageUrl":"images/marinara.png"}, {"id":4,"pizzaName":"Carciofi","description":"Pizza with tomato, fresh mozzarella and artichokes","price":8.8,"imageUrl":"images/marinara.png"}, {"id":5,"pizzaName":"Bufala","description":"Pizza with tomato, buffalo mozzarella and basil","price":12.5,"imageUrl":"images/marinara.png"}]
```

.Praktikum 3: Saving data simply with SharedPreferences

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

```
PS D:\Belajar\Mobile Programming\versaoktavian_store_data> flutter pub add shared_preferences
"shared_preferences" is already in "dependencies". Will try to update the constraint.
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 (1.4.0 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)
  flutter_lints 4.0.0 (5.0.0 available)
  leak_tracker 10.0.5 (10.0.8 available)
```

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

```
PROBLEMS 1 OUTPUT TERMINAL ... pwsh + - [ ] [X] ... ^ X

Try `flutter pub outdated` for more information.
PS D:\Belajar\Mobile Programming\versaoktavian_store_data> 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 (1.4.0 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)
  flutter_lints 4.0.0 (5.0.0 available)
```

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

```
2 import 'package:flutter/material.dart';
3 | import 'package:shared_preferences/shared_preferences.dart';
4 import './pizza.dart';
```

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

```
35 |
36 | int appCounter = 0;
```

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

```
63  
64 | Future readAndWritePreference() async {
```

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

```
65 | 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 naikan nilainya:

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

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

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

9. Memperbarui nilai status `appCounter`:

```
69 |         setState() {  
70 |             appCounter = appCounter;  
71 |         });  
72 |     }
```

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

```
    readAndWritePreference();  
}
```

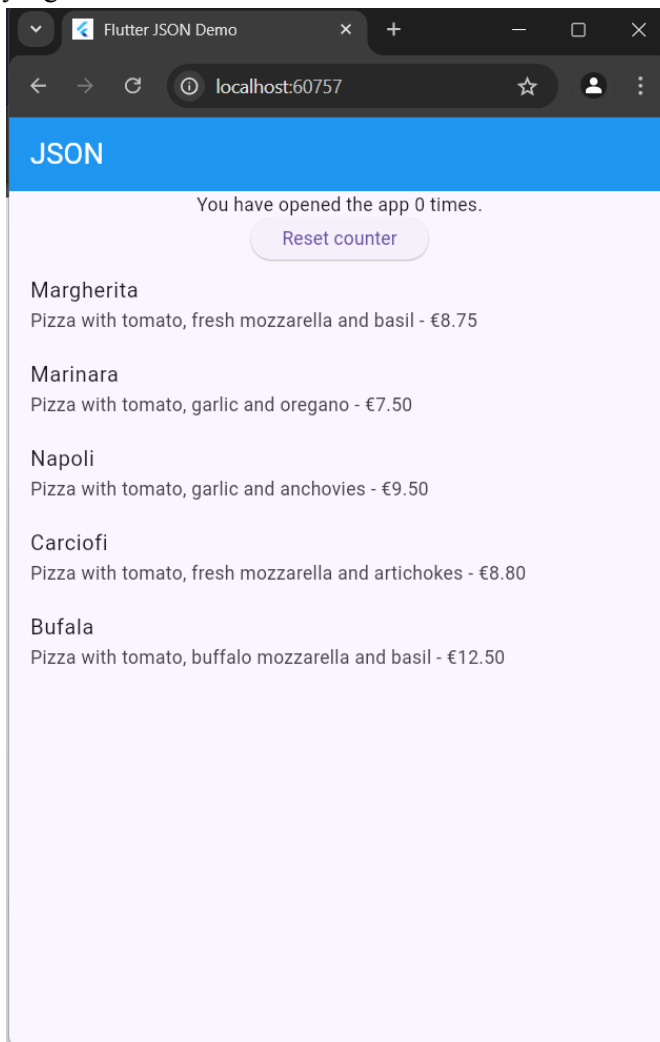
11. Dalam metode `build`, tambahkan kode berikut ini di dalam widget `Container`:

```

92   child: Column(
93     mainAxisAlignment: MainAxisAlignment.spaceEvenly,
94     children: [
95       Text(
96         'You have opened the app $appCounter times.', // Text
97       ),
98       ElevatedButton(
99         onPressed: () async {
100           SharedPreferences prefs = await SharedPreferences.getInstance();
101           await prefs.setInt('appCounter', 0);
102           setState(() {
103             appCounter = 0;
104           });
105         },
106         child: Text('Reset counter'), // Use 'const' with the const
107       ), // ElevatedButton

```

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



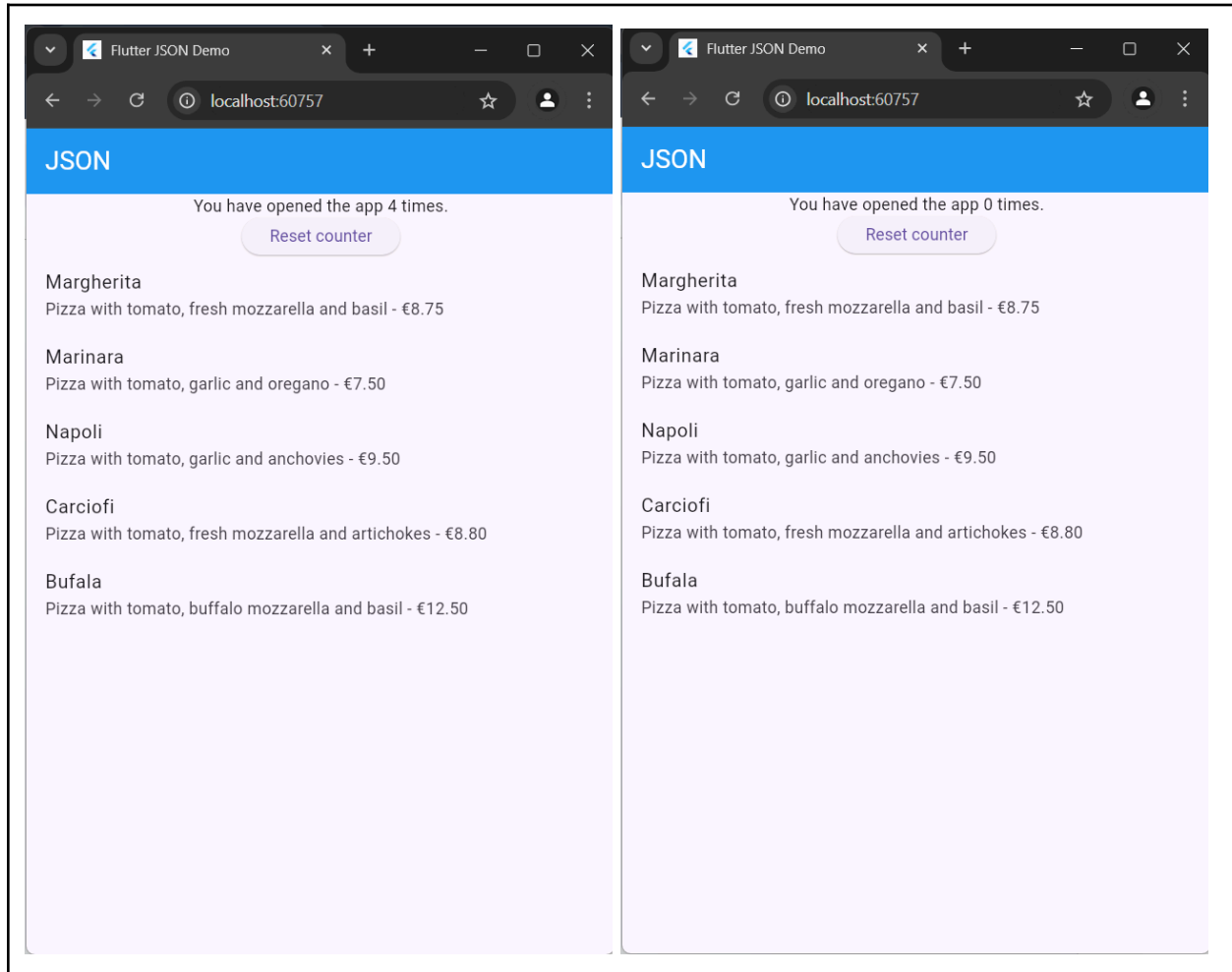
13. Tambahkan metode baru ke kelas `_MyHomePageState` yang disebut `deletePreference()`, yang akan menghapus nilai yang disimpan:

```
73
74   Future deletePreference() async {
75     SharedPreferences prefs = await SharedPreferences.getInstance();
76     await prefs.clear();
77     setState(() {
78       appCounter = 0;
79     });
80   }
81
```

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

```
104   // You have opened the app $appcounter times. ), // Text
105   ElevatedButton(
106     onPressed: () {
107       deletePreference();
108     },
```

15. Jalankan aplikasi lagi. Sekarang, saat Anda menekan tombol Reset penghitung, nilai `appCounter` akan dihapus



Praktikum 4: Accessing the filesystem, part 1: path_provider

Buatlah project flutter baru dengan nama path_provider

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

```

PS D:\Belajar\Mobile Programming\versaoktavian_store_data> flutter pub add path_provider
Resolving dependencies...
Downloading packages... (1.8s)
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  characters 1.3.0 (1.4.0 available)
  clock 1.1.1 (1.1.2 available)
> collection 1.19.0 (was 1.18.0) (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
  flutter_lints 4.0.0 (5.0.0 available)
> leak_tracker 10.0.7 (was 10.0.5) (10.0.8 available)
> leak_tracker_flutter_testing 3.0.8 (was 3.0.5) (3.0.9 available)

```

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

```

1 import 'dart:convert';
2 import 'package:flutter/material.dart';
3 | import 'package:path_provider/path_provider.dart';
4 import 'package:shared_preferences/shared_preferences.dart';
5 import './pizza.dart';

```

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

```

> android 31 class _MyHomePageState extends State<MyHomePage> {
> build 32 String documentsPath = '';
> ios 33 String tempPath = '';
34

```

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

```

86 Future getPaths() async {
87   final docDir = await getApplicationDocumentsDirectory();
88   final tempDir = await getTemporaryDirectory();
89   setState(() {
90     documentsPath = docDir.path;
91     tempPath = tempDir.path;
92   });
93 }

```

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

```

34
35   @override
36   void initState() {
37     super.initState();
38     getPaths();
39   }
40

```

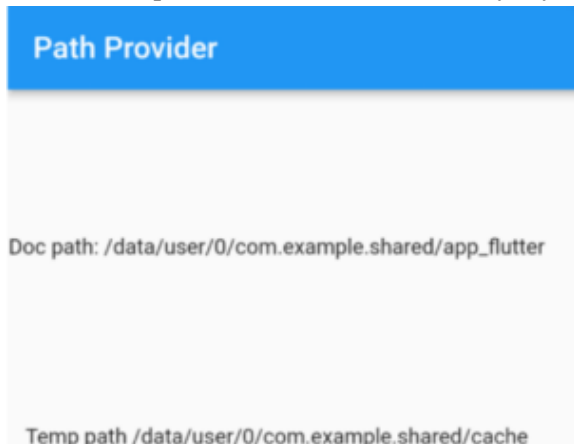
6. Pada metode `build_MyHomePageState`, buat UI dengan dua widget Teks yang menunjukkan path yang diambil:

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Path Provider'),
    ), // AppBar
    body: Column(
      mainAxisAlignment: MainAxisAlignment.spaceEven
      children: [
        Text('Doc path: $documentsPath'),
        Text('Temp path: $tempPath'),
      ],
    ), // Column
  ); // Scaffold
}

```

7. Jalankan aplikasi. Anda akan melihat layar yang terlihat seperti berikut ini:



Praktikum 5: Accessing the filesystem, part 2: Working with directories

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:

```
101 Future<bool> writeFile() async {  
102   try {  
103     await myFile.writeAsString('Margherita, Capricci  
104     return true;  
105   } catch (e) {}  
106   return false;  
107 }  
108
```

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

```
52 getPaths().then((_) {  
53   myFile = File('$documentsPath/pizzas.txt');  
54   writeFile();  
55 });  
56
```

5. Buat metode untuk membaca file:

```

110     Future<bool> readFile() async {
111         try {
112             String fileContent = await myFile.readAsString()
113             setState(() {
114                 fileText = fileContent;
115             });
116             return true;
117         } catch (e) {
118             return false;
119         }
120     }
121

```

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:

```

163         Text('Doc path: $documentsPath'),
164         Text('Temp path: $tempPath'),
165         ElevatedButton(
166             child: const Text('Read File'),
167             onPressed: () => readFile(),
168         ), // ElevatedButton
169         Text(fileText),
170     ],
171 ), // Column
172 ), // Center
173 ), // Container
174 ); // Scaffold
175 }

```

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: Using secure storage to store data

1. Tambahkan flutter_secure_storage ke proyek Anda, dengan mengetik:

```
PS D:\Belajar\Mobile Programming\ersaoktavian_store_data> flutter pub add flutter_secure_storage
Resolving dependencies...
Downloading packages... (31.8s)
  async 2.11.0 (2.12.0 available)
  boolean_selector 2.1.1 (2.1.2 available)
  character 1.3.0 (1.4.0 available)
  clock 1.1.1 (1.1.2 available)
  collection 1.19.0 (1.19.1 available)
  fake_async 1.3.1 (1.3.2 available)
  flutter_lints 4.0.0 (5.0.0 available)
  Source: Dart
  flutter pub get --no-example: running...
  Cancel
```

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

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

void main() async {
```

```

WidgetsFlutterBinding.ensureInitialized();
runApp(const MyApp());
}

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

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Path Provider',
      theme: ThemeData(
        useMaterial3: true,
      ),
      home: const MyHomePage(title: 'Path Provider'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key, required this.title});

  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  String documentsPath = '';
  String tempPath = '';
  late File myFile;
  String fileText = '';

  Future getPaths() async{

```



```

final docDir = await getApplicationDocumentsDirectory();
final tempDir = await getTemporaryDirectory();
setState(() {
  documentsPath = docDir.path;
  tempPath = tempDir.path;
});
}
Future<bool> writeFile() async{
  try{
    await myFile.writeAsString('Margherita, Capricciosa, Napoli');
    return true;
  }catch(e){
    return false;
  }
}
Future<bool> readFile() async{
  try{
    String fileContent = await myFile.readAsString();
    setState(() {
      fileText =fileContent;
    });
    return true;
  }catch(e){
    return false;
  }
}

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

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: const Text('Path Provider')),
    body: Column(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      children: [
        Text('Doc path: $documentsPath'),
        Text('Temp path $tempPath'),
        ElevatedButton(
          onPressed: () => readFile(),
          child: const Text('Read File')
        ),
        Text(fileText),
      ],
    ),
  );
}

```

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

```

import 'package:path_provider/path_provider.dart';

```

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

```

35 String documentsPath = '';
36 String tempPath = '';
37 late File myFile;
38 String fileText = '';

```

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

```
8 Future<bool> writeFile() async{
9   try{
10    await myFile.writeAsString('Margherita, Capricci
11    return true;
12  }
```

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

```
7 }
8 Future<bool> writeFile() async{
```

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

```
97
98 Future<String> readFromSecureStorage() async {
99   String secret = await storage.read(key: myKey) ??
100   return secret;
101 }
```

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

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:

