

LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE



Oleh:

Farlyhaydy H.Djalil NIM. 2210817210006

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
2024**

LEMBAR PENGESAHAN

LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

Laporan Praktikum Pemrograman Mobile

Modul 1 : Android Basic with Kotlin

Modul 2 : Android Layout

Modul 3 : Android Navigation

Modul 4 : Connect to the internet

Modul 5 : Android UI Design

ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Farlyhaydy H.Djalil

NIM : 2210817210006

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Asandy Putra
NIM. 2110817310002

Muti'a Maulida, S.Kom., M.T.I.
NIP. 198810272019032013

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI.....	3
DAFTAR GAMBAR	5
DAFTAR TABEL.....	7
MODUL 1 : ANDROID BASIC WITH KOTLIN	9
SOAL 1	9
A. Source Code	11
B. Output Program	16
C. Pembahasan	18
D. Tautan Git.....	25
MODUL 2 : ANDROID LAYOUT	27
SOAL 1	27
A. Source Code	28
B. Output Program	33
C. Pembahasan	35
D. Tautan Git.....	44
MODUL 3 : ANDROID NAVIGATION	45
SOAL 1	45
A. Source Code	47
B. Output Program	61
C. Pembahasan	63
D. Tautan Git.....	85
MODUL 4 : CONNECT TO THE INTERNET	86
SOAL 1	86
A. Source Code	88
B. Output Program	107
C. Pembahasan	108

D. Tautan Git.....	119
MODUL 5 : ANDROID UI DESIGN.....	120
SOAL 1	120
A. Source Code	121
B. Output Program	152
C. Pembahasan	154
D. Tautan Git.....	181

DAFTAR GAMBAR

Modul 1 : Android Basic with Kotlin

Gambar 1. Tampilan Awal Aplikasi	9
Gambar 2. Tampilan Dadu Setelah Di Roll	10
Gambar 3. Tampilan Roll Dadu Double	11
Gambar 4. Screenshot Hasil Jawaban Soal 1 yang ke 1	16
Gambar 5. Screenshot Hasil Jawaban Soal 1 yang ke 2	17
Gambar 6. Screenshot Hasil Jawaban Soal 1 yang ke 3	18

Modul 2 : Android Layout

Gambar 7. Tampilan Awal Aplikasi	27
Gambar 8. Tampilan Aplikasi Setelah Dijalankan.....	28
Gambar 9. Screenshot Hasil Jawaban Soal 1 yang ke 1	33
Gambar 10. Screenshot Hasil Jawaban Soal 1 yang ke 2	34

Modul 3 : Android Navigation

Gambar 11. Contoh pertama	45
Gambar 12. Contoh kedua.....	46
Gambar 13. Contoh Ketiga	47
Gambar 14. Screenshot Hasil Jawaban Soal 1 yang ke 1	61
Gambar 15. Screenshot Hasil Jawaban Soal 1 yang ke 2	62
Gambar 16. Screenshot Hasil Jawaban Soal 1 yang ke 3	63

Modul 4 : Connect to the internet

Gambar 17. Contoh gambar pertama	87
Gambar 18. Contoh gambar kedua.....	87
Gambar 19. Screenshot Hasil Jawaban Soal 1 yang ke 1	107
Gambar 20. Screenshot Hasil Jawaban Soal 1 yang ke 2	108

Modul 5 : Android UI Design

Gambar 21. Contoh gambar pertama	120
Gambar 22. Contoh gambar kedua.....	121
Gambar 23. Screenshot Hasil Jawaban Soal 1 tampilan login.....	152

Gambar 24. Screenshot Hasil Jawaban Soal 1 tampilan di halaman utama.....	153
Gambar 25. Screenshot Hasil Jawaban Soal 1 tampilan menu	154

DAFTAR TABEL

Modul 1 : Android Basic with Kotlin

Tabel 1. Source Code Soal 1 yang Kotlin	14
Tabel 2. Source Code Soal 1 yang XML	15

Modul 2 : Android Layout

Tabel 3. Source Code Soal 1 yang Kotlin	29
Tabel 4. Source Code Soal 1 yang XML	32

Modul 3 : Android Navigation

Tabel 5. Source Code Soal 1 Kotlin yang pertama	50
Tabel 6. Source Code Soal 1 Kotlin yang kedua	52
Tabel 7. Source Code Soal 1 Kotlin yang ketiga	54
Tabel 8. Source Code Soal 1 Kotlin yang keempat	55
Tabel 9. Source Code Soal 1 XML yang pertama	59
Tabel 10. Source Code Soal 1 XML yang kedua.....	60

Modul 4 : Connect to the internet

Tabel 11. Source Code Soal 1 Kotlin yang pertama	89
Tabel 12. Source Code Soal 1 Kotlin yang kedua	91
Tabel 13. Source Code Soal 1 Kotlin yang ketiga	92
Tabel 14. Source Code Soal 1 Kotlin yang keempat	95
Tabel 15. Source Code Soal 1 Kotlin yang kelima	96
Tabel 16. Source Code Soal 1 Kotlin yang keenam.....	99
Tabel 17. Source Code Soal 1 Kotlin yang ketujuh.....	100
Tabel 18. Source Code Soal 1 XML yang pertama	101
Tabel 19. Source Code Soal 1 XML yang kedua.....	103
Tabel 20. Source Code Soal 1 XML yang ketiga.....	105
Tabel 21. Source Code Soal 1 XML yang keempat.....	106

Modul 5 : Android UI Design

Tabel 22. Source Code Soal 1 Kotlin yang pertama	122
Tabel 23. Source Code Soal 1 Kotlin yang kedua	123

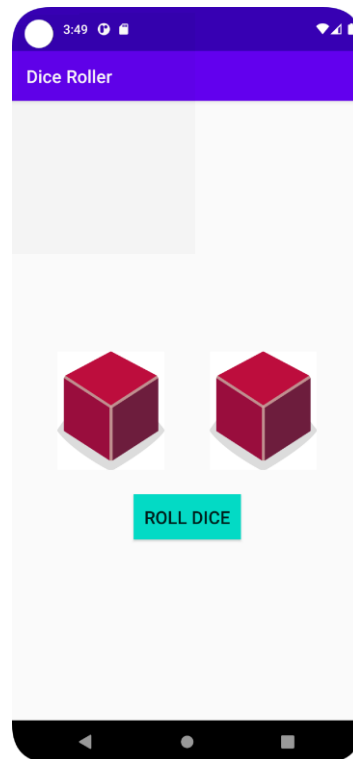
Tabel 24. Source Code Soal 1 Kotlin yang ketiga	124
Tabel 25. Source Code Soal 1 Kotlin yang keempat	126
Tabel 26. Source Code Soal 1 Kotlin yang kelima	127
Tabel 27. Source Code Soal 1 Kotlin yang keenam.....	129
Tabel 28. Source Code Soal 1 Kotlin yang ketujuh.....	130
Tabel 29. Source Code Soal 1 Kotlin yang kedelapan.....	130
Tabel 30. Source Code Soal 1 Kotlin yang kesembilan.....	132
Tabel 31. Source Code Soal 1 Kotlin yang kesepuluh.....	133
Tabel 32. Source Code Soal 1 Kotlin yang kesebelas.....	134
Tabel 33. Source Code Soal 1 Kotlin yang kedua belas	137
Tabel 34. Source Code Soal 1 Kotlin yang ketiga belas	137
Tabel 35. Source Code Soal 1 XML yang pertama	139
Tabel 36. Source Code Soal 1 XML yang kedua.....	142
Tabel 37. Source Code Soal 1 XML yang ketiga.....	142
Tabel 38. Source Code Soal 1 XML yang keempat.....	144
Tabel 39. Source Code Soal 1 XML yang kelima	146
Tabel 40. Source Code Soal 1 XML yang keenam	147
Tabel 41. Source Code Soal 1 XML yang ketujuh	149
Tabel 42. Source Code Soal 1 XML yang kedelean	151

MODUL 1 : ANDROID BASIC WITH KOTLIN

SOAL 1

Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

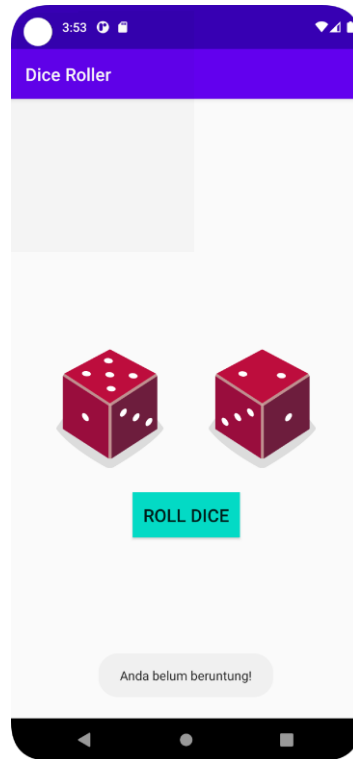
1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



Gambar 1. Tampilan Awal Aplikasi

Gambar 1 Tampilan Awal Aplikasi

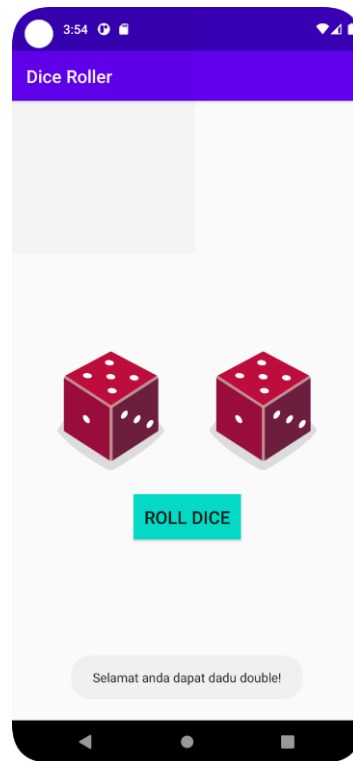
2. Setelah user menekan tombol “Roll Dice” maka masing-masing dadu akan memunculkan sisi dadu masing-masing dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2 maka akan menampilkan pesan “Anda belum beruntung!” seperti dapat dilihat pada Gambar 2.



Gambar 2. Tampilan Dadu Setelah Di Roll

Gambar 2 Tampilan Dadu Setelah Di Roll

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan “Selamat anda dapat dadu double!” Seperti dapat dilihat pada Gambar 3.
4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project**. Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.
5. Untuk gambar dadu dapat didownload pada link berikut:
https://drive.google.com/u/0/uc?id=147HT2IIH5qin3z5ta7H9y2N_5OMW81Ll&export=download



Gambar 3. Tampilan Roll Dadu Double

Gambar 3 Tampilan Roll Dadu Double

A. Source Code

```

1 package com.example.RoleDice
2
3 import android.graphics.Color
4 import android.graphics.PorterDuff
5 import android.widget.TextView
6 import android.widget.Toast
7 import androidx.appcompat.app.AppCompatActivity
8 import android.os.Bundle
9 import android.widget.Button
10 import android.widget.ImageView
11 import com.example.tugasroledice.R
12
13 class MainActivity : AppCompatActivity() {
14
15     override fun onCreate(savedInstanceState:
Bundle?) {

```

```

16         super.onCreate(savedInstanceState)
17         setContentView(R.layout.activity_main)
18
19         val rollButton: Button =
20     findViewById(R.id.let_roll)
21         rollButton.setOnClickListener { roll() }
22     }
23
24     private fun roll() {
25         var dadulagi1 = diceroll1()
26         var dadulagi2 = diceroll2()
27
28         if (dadulagi1 == dadulagi2) {
29             showCustomToast("Selamat anda dapat
30     dadu double!", Color.parseColor("#F0F0F0"),
31     Color.BLACK)
32         } else {
33             showCustomToast("Anda belum
34     beruntung!", Color.parseColor("#F0F0F0"),
35     Color.BLACK)
36         }
37     }
38
39     private fun showCustomToast(message: String,
40     backgroundColor: Int, textColor: Int) {
41         val toast = Toast.makeText(this, message,
42     Toast.LENGTH_SHORT)
43         val view = toast.view
44
45         // Mengatur warna latar belakang Toast
46
47         view?.background?.setColorFilter(backgroundColor,
48     PorterDuff.Mode.SRC_IN)
49
50         // Mendapatkan TextView dari Toast
51         sehingga dapat diedit
52         val text =
53     view?.findViewById<TextView>(android.R.id.message)
54         text?.setTextColor(textColor)
55
56         toast.show()
57     }
58 }

```

```

48     private fun diceroll1(): Int {
49         val dice1 = Dice(6)
50         val diceRoll1 = dice1.roll()
51
52         val diceImage1: ImageView =
53 findViewById(R.id.dice_image1)
54         val drawableResource1 = when (diceRoll1) {
55             1 -> R.drawable.dice_1
56             2 -> R.drawable.dice_2
57             3 -> R.drawable.dice_3
58             4 -> R.drawable.dice_4
59             5 -> R.drawable.dice_5
60             else -> R.drawable.dice_6
61         }
62
63 diceImage1.setImageResource(drawableResource1)
64 diceImage1.contentDescription =
65 diceRoll1.toString()
66         return diceRoll1
67     }
68
69     private fun diceroll2(): Int {
70         val dice2 = Dice(6)
71         val diceRoll2 = dice2.roll()
72
73         val diceImage2: ImageView =
74 findViewById(R.id.dice_image2)
75         val drawableResource2 = when (diceRoll2) {
76             1 -> R.drawable.dice_1
77             2 -> R.drawable.dice_2
78             3 -> R.drawable.dice_3
79             4 -> R.drawable.dice_4
80             5 -> R.drawable.dice_5
81             else -> R.drawable.dice_6
82         }
83
84 diceImage2.setImageResource(drawableResource2)
85 diceImage2.contentDescription =
86 diceRoll2.toString()
87         return diceRoll2
88     }

```

```

85 }
86
87 class Dice(private val numSides: Int) {
88
89     /**
90      * Lakukan pelemparan dadu secara acak dan
91      * mengembalikan hasilnya.
92      */
93     fun roll(): Int {
94         return (1..numSides).random()
95     }
96 }

```

Tabel 1. Source Code Soal 1 yang Kotlin

```

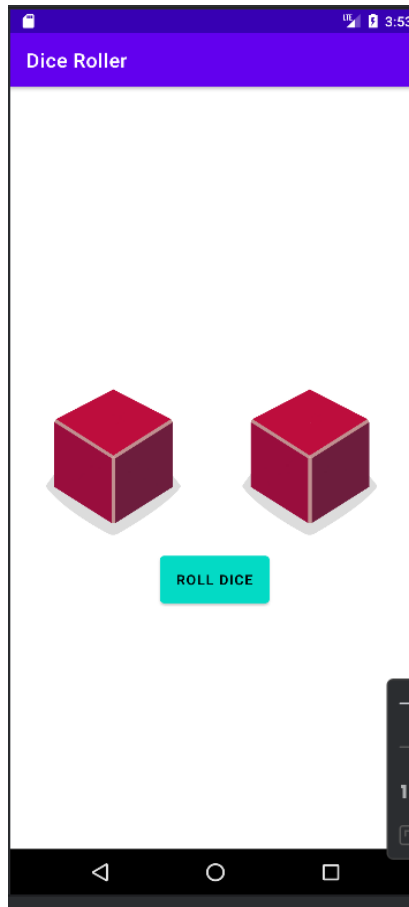
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3  xmlns:android="http://schemas.android.com/apk/res/an
4  droid"
5      xmlns:app="http://schemas.android.com/apk/res-
6  auto"
7      xmlns:tools="http://schemas.android.com/tools"
8      android:layout_width="match_parent"
9      android:layout_height="match_parent"
10     android:layout_gravity="center_vertical"
11     android:orientation="vertical"
12
13     tools:context="com.example.RoleDice.MainActivity">
14
15     <ImageView
16         android:id="@+id/dice_image1"
17         android:layout_width="150dp"
18         android:layout_height="150dp"
19         android:layout_marginStart="28dp"
20         android:layout_marginTop="312dp"
21         android:src="@drawable/empty_dice"
22
23     app:layout_constraintStart_toStartOf="parent"
24     app:layout_constraintTop_toTopOf="parent" />
25
26     <ImageView
27         android:id="@+id/dice_image2"
28         android:layout_width="150dp"

```

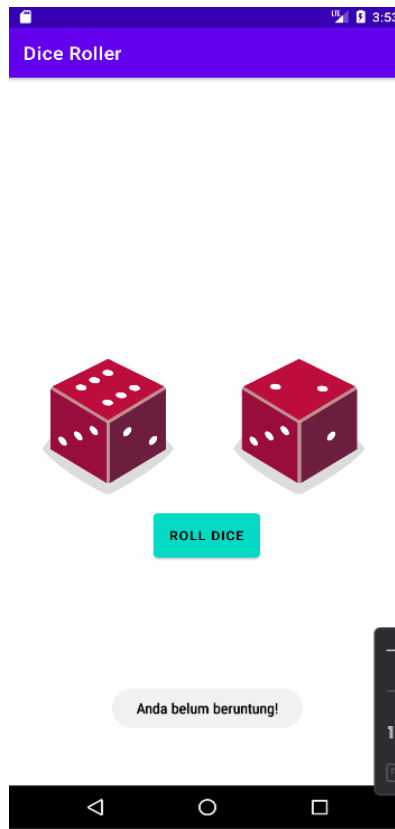
29	android:layout_height="150dp"
30	android:layout_marginTop="312dp"
31	android:layout_marginEnd="36dp"
32	android:src="@drawable/empty_dice"
33	app:layout_constraintEnd_toEndOf="parent"
34	app:layout_constraintTop_toTopOf="parent" />
35	
36	<Button
37	android:id="@+id/let_roll"
38	android:layout_width="wrap_content"
39	android:layout_height="60dp"
40	android:layout_marginTop="16dp"
41	android:backgroundTint="#FF03DAC5"
42	android:text="@string/roll_dice"
43	android:textColor="#000000"
44	
45	app:layout_constraintBaseline_toBottomOf="parent"
46	app:layout_constraintEnd_toEndOf="parent"
47	app:layout_constraintHorizontal_bias="0.498"
48	
49	
50	
51	app:layout_constraintStart_toStartOf="parent"
52	
53	app:layout_constraintTop_toBottomOf="@id/dice_image1
	" />
54	
55	</androidx.constraintlayout.widget.ConstraintLayout>

Tabel 2. Source Code Soal 1 yang XML

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 yang ke 1



Gambar 5. Screenshot Hasil Jawaban Soal 1 yang ke 2



Gambar 6. Screenshot Hasil Jawaban Soal 1 yang ke 3

C. Pembahasan

Untuk file kotlin

Pada baris ke 1, `package com.example.RoleDice`: Ini adalah deklarasi paket yang berisi semua kelas dalam file. Dalam hal ini, kelas-kelas akan dimasukkan ke dalam paket `com.example.RoleDice`.

Pada baris ke 3-11, `import statements`: Ini mengimpor kelas-kelas yang diperlukan dari paket `android`, `androidx.appcompat.app`, dan paket lokal `com.example.tugasroledice.R`.

Pada baris ke 13, `class MainActivity : AppCompatActivity() {}`: Ini mendefinisikan kelas `MainActivity`, yang merupakan turunan dari kelas `AppCompatActivity`, yang berarti aktivitas ini akan menggunakan fitur-fitur dari `Support Library` untuk kompatibilitas yang lebih baik dengan versi `Android` yang lebih lama.

Pada baris ke 15, override fun onCreate(savedInstanceState: Bundle?) {: Ini adalah fungsi yang dipanggil saat aktivitas dibuat. Fungsi ini mengambil savedInstanceState sebagai argumen, yang berisi data yang mungkin sudah disimpan dari aktivitas sebelumnya.

Pada baris ke 16, super.onCreate(savedInstanceState): Memanggil implementasi kelas induk onCreate() untuk melakukan inisialisasi dasar aktivitas.

Pada baris ke 17, setContentView(R.layout.activity_main): Menetapkan tata letak XML yang didefinisikan dalam file activity_main.xml sebagai tata letak untuk aktivitas.

Pada baris ke 19, val rollButton: Button = findViewById(R.id.let_roll): Mencari tombol dengan ID let_roll dalam tata letak dan menentukannya ke variabel rollButton.

Pada baris ke 20, rollButton.setOnClickListener { roll() }: Menetapkan listener onClick untuk tombol rollButton yang akan memanggil fungsi roll() saat tombol diklik.

Pada baris ke 23, private fun roll() {: Mendefinisikan fungsi roll() yang akan dipanggil saat tombol dilepas.

Pada baris ke 24, var dadulagi1 = diceroll1(): Memanggil fungsi diceroll1() untuk melempar dadu pertama dan menyimpan hasilnya di variabel dadulagi1.

Pada baris ke 25, var dadulagi2 = diceroll2(): Memanggil fungsi diceroll2() untuk melempar dadu kedua dan menyimpan hasilnya di variabel dadulagi2.

Pada baris ke 27, if (dadulagi1 == dadulagi2) {: Ini adalah awal dari pernyataan kontrol if. Ini mengevaluasi apakah nilai dadulagi1 sama dengan nilai dadulagi2.

Pada baris ke 28, showCustomToast("Selamat anda dapat dadu double!", Color.parseColor("#F0F0F0"), Color.BLACK): Jika kondisi di atas benar (yaitu, kedua dadu memiliki nilai yang sama), maka pesan "Selamat anda dapat dadu double!" akan ditampilkan menggunakan toast kustom yang telah didefinisikan sebelumnya dengan latar belakang berwarna #F0F0F0 dan teks berwarna hitam.

Pada baris ke 29, else {: Ini adalah bagian else dari pernyataan if. Jika kondisi di atas tidak terpenuhi (yaitu, kedua dadu memiliki nilai yang berbeda), maka pernyataan dalam blok ini akan dieksekusi.

Pada baris ke 30, showCustomToast("Anda belum beruntung!", Color.parseColor("#F0F0F0"), Color.BLACK): Jika kondisi di atas tidak terpenuhi, maka pesan "Anda belum beruntung!" akan ditampilkan menggunakan toast kustom dengan latar belakang berwarna #F0F0F0 dan teks berwarna hitam.

Pada baris ke 31, `}`: Ini menutup blok `else` dan juga menutup blok `if`. Ini menandakan akhir dari pernyataan kontrol `if-else`.

Pada baris ke 34, `private fun showCustomToast(message: String, backgroundColor: Int, textColor: Int) { ... }`: Mendefinisikan fungsi untuk menampilkan toast kustom dengan pesan, warna latar belakang, dan warna teks yang disediakan.

Pada baris ke 35, `val toast = Toast.makeText(this, message, Toast.LENGTH_SHORT)`: Membuat objek toast dengan pesan yang diberikan.

Pada baris ke 36, `val view = toast.view`: Mendapatkan tampilan yang terkait dengan toast. Tampilan ini dapat diedit untuk menyesuaikan tampilan toast.

Pada baris ke 39, `view?.background?.setColorFilter(backgroundColor, PorterDuff.Mode.SRC_IN)`: Mengatur warna latar belakang toast sesuai dengan warna yang diberikan.

Pada baris ke 42, `val text = view?.findViewById<TextView>(android.R.id.message)`: Mendapatkan `TextView` dari toast untuk dapat mengatur warna teksnya.

Pada baris ke 43, `text?.setTextColor(textColor)`: Mengatur warna teks toast sesuai dengan warna yang diberikan.

Pada baris ke 45, `toast.show()`: Menampilkan toast.

Pada baris ke 48, `private fun diceroll1(): Int { ... }`: Mendefinisikan fungsi bernama `diceroll1()` yang mengembalikan nilai bertipe data `Int`. Fungsi ini akan menghasilkan hasil lemparan dadu pertama.

Pada baris ke 49, `val dice1 = Dice(6)`: Membuat objek `dice1` dari kelas `Dice` dengan mengirimkan jumlah sisi dadu, yaitu 6, sebagai parameter. Ini merupakan inisialisasi dadu pertama.

Pada baris ke 50, `val diceRoll1 = dice1.roll()`: Memanggil metode `roll()` dari objek `dice1` untuk melakukan lemparan dadu dan menyimpan hasilnya ke dalam variabel `diceRoll1`.

Pada baris ke 52, `val diceImage1: ImageView = findViewById(R.id.dice_image1)`: Mencari `ImageView` dengan ID `dice_image1` di tata letak yang terkait dengan aktivitas saat ini dan menentukannya ke dalam variabel `diceImage1`. `ImageView` ini akan digunakan untuk menampilkan gambar dadu pertama.

Pada baris ke 53-59, `val drawableResource1 = when (diceRoll1) { ... }`: Menggunakan ekspresi `when` untuk memilih `drawable` yang sesuai dengan hasil lemparan dadu

pertama. Setiap nilai `diceRoll1` akan diuji dan sesuai `drawable` akan dipilih berdasarkan nilai dadu.

Pada baris ke 62, `diceImage1.setImageResource(drawableResource1)`: Menetapkan gambar dadu yang sesuai ke `ImageView` `diceImage1` menggunakan metode `setImageResource()`. Ini akan menampilkan gambar dadu yang sesuai dengan hasil lemparan.

Pada baris ke 63, `diceImage1.contentDescription = diceRoll1.toString()`: Mengatur deskripsi konten untuk `ImageView` `diceImage1` agar berisi nilai dadu sebagai teks. Ini adalah praktik yang baik untuk aksesibilitas, karena memungkinkan pengguna yang menggunakan pembaca layar untuk mengetahui nilai dadu.

Pada baris ke 64, `return diceRoll1`: Mengembalikan nilai dadu pertama sebagai hasil dari fungsi `diceroll1()`.

Pada baris ke 67, `private fun diceroll2(): Int {`: Mendefinisikan fungsi bernama `diceroll2()` yang mengembalikan nilai bertipe data `Int`. Fungsi ini akan menghasilkan hasil lemparan dadu kedua.

Pada baris ke 68, `val dice2 = Dice(6)`: Membuat objek `dice2` dari kelas `Dice` dengan mengirimkan jumlah sisi dadu, yaitu 6, sebagai parameter. Ini merupakan inisialisasi dadu kedua.

Pada baris ke 69, `val diceRoll2 = dice2.roll()`: Memanggil metode `roll()` dari objek `dice2` untuk melakukan lemparan dadu dan menyimpan hasilnya ke dalam variabel `diceRoll2`.

Pada baris ke 71, `val diceImage2: ImageView = findViewById(R.id.dice_image2)`: Mencari `ImageView` dengan ID `dice_image2` di tata letak yang terkait dengan aktivitas saat ini dan menetakannya ke dalam variabel `diceImage2`. `ImageView` ini akan digunakan untuk menampilkan gambar dadu kedua.

Pada baris ke 72-78, `val drawableResource2 = when (diceRoll2) { ... }`: Menggunakan ekspresi `when` untuk memilih `drawable` yang sesuai dengan hasil lemparan dadu kedua. Setiap nilai `diceRoll2` akan diuji dan sesuai `drawable` akan dipilih berdasarkan nilai dadu.

Pada baris ke 81, `diceImage2.setImageResource(drawableResource2)`: Menetapkan gambar dadu yang sesuai ke `ImageView` `diceImage2` menggunakan metode `setImageResource()`. Ini akan menampilkan gambar dadu yang sesuai dengan hasil lemparan.

Pada baris ke 82, `diceImage2.contentDescription = diceRoll2.toString()`: Mengatur deskripsi konten untuk `ImageView` `diceImage2` agar berisi nilai dadu sebagai teks. Ini adalah praktik yang baik untuk aksesibilitas, karena memungkinkan pengguna yang menggunakan pembaca layar untuk mengetahui nilai dadu.

Pada baris ke 83, `return diceRoll2`: Mengembalikan nilai dadu kedua sebagai hasil dari fungsi `diceroll2()`.

Pada baris ke 84, `}`: Tutup dari fungsi `diceroll2()`. Menandakan akhir dari definisi fungsi tersebut.

Pada baris ke 87, `class Dice(private val numSides: Int) {}`: Mendefinisikan kelas `Dice`. Kelas ini memiliki satu properti `numSides` yang merupakan jumlah sisi dadu. Kata kunci `private` pada deklarasi properti mengindikasikan bahwa properti ini hanya dapat diakses di dalam kelas `Dice` itu sendiri.

Pada baris ke 92, `fun roll(): Int {}`: Mendefinisikan fungsi `roll()` yang mengembalikan nilai bertipe data `Int`. Fungsi ini bertanggung jawab untuk melakukan pelemparan dadu dan mengembalikan hasilnya.

Pada baris ke 93, `return (1..numSides).random()`: Ini adalah isi dari fungsi `roll()`. Fungsi ini menggunakan ekspresi `(1..numSides)` untuk membuat rentang angka dari 1 hingga `numSides`, kemudian memanggil metode `random()` pada rentang tersebut untuk memilih angka secara acak. Hasil dari pemilihan tersebut kemudian dikembalikan sebagai hasil pelemparan dadu.

Pada baris ke 94, `}`: Tutup dari fungsi `roll()`. Menandakan akhir dari definisi fungsi tersebut.

Pada baris ke 95, `}`: Tutup dari kelas `Dice`. Menandakan akhir dari definisi kelas tersebut.

Untuk file XML

Pada baris ke 1, `<?xml version="1.0" encoding="utf-8"?>`: Mendefinisikan versi XML dan jenis encoding yang digunakan dalam file.

Pada baris ke 2, `<androidx.constraintlayout.widget.ConstraintLayout>`: Merupakan root element dari layout file yang menunjukkan bahwa kita menggunakan `ConstraintLayout` sebagai layout container untuk menempatkan elemen-elemen UI.

Pada baris ke 2, `xmlns:android="http://schemas.android.com/apk/res/android"`: Mendefinisikan namespace yang digunakan untuk elemen-elemen yang berasal dari Android.

Pada baris ke 3, `xmlns:app="http://schemas.android.com/apk/res-auto"`: Mendefinisikan namespace yang digunakan untuk atribut-atribut yang berasal dari library appcompat atau dari library support lainnya.

Pada baris ke 4, `xmlns:tools="http://schemas.android.com/tools"`: Mendefinisikan namespace yang digunakan untuk atribut-atribut yang hanya diperlukan dalam waktu kompilasi dan tidak akan dimasukkan ke dalam aplikasi saat runtime.

Pada baris ke 5, `android:layout_width="match_parent"`: Mengatur lebar layout menjadi sama dengan lebar parent layout-nya.

Pada baris ke 6, `android:layout_height="match_parent"`: Mengatur tinggi layout menjadi sama dengan tinggi parent layout-nya.

Pada baris ke 7, `android:layout_gravity="center_vertical"`: Mengatur gravity dari layout, dalam hal ini diatur ke tengah secara vertikal.

Pada baris ke 8, `android:orientation="vertical"`: Mengatur orientasi dari layout, dalam hal ini diatur menjadi vertikal.

Pada baris ke 9, `tools:context="com.example.RoleDice.MainActivity"`: Memberikan konteks dari layout untuk keperluan tampilan editor atau tools.

Pada baris ke 11, `<ImageView`: Mendefinisikan sebuah ImageView, yaitu elemen untuk menampilkan gambar atau grafis.

Pada baris ke 12, `android:id="@+id/dice_image1"`: Memberikan identifier (ID) unik untuk ImageView ini agar dapat diidentifikasi dan diakses dalam kode Java atau Kotlin.

Pada baris ke 13, `android:layout_width="150dp"`: Menetapkan lebar ImageView menjadi 150dp (density-independent pixels), sehingga gambar akan memiliki lebar tetap.

Pada baris ke 14, `android:layout_height="150dp"`: Menetapkan tinggi ImageView menjadi 150dp, sehingga gambar akan memiliki tinggi tetap.

Pada baris ke 15, `android:layout_marginStart="28dp"`: Menetapkan margin (jarak) dari sisi kiri (atau sisi awal) ImageView sebesar 28dp.

Pada baris ke 16, `android:layout_marginTop="312dp"`: Menetapkan margin dari sisi atas ImageView sebesar 312dp.

Pada baris ke 17, `android:src="@drawable/empty_dice"`: Menetapkan sumber gambar untuk ditampilkan di ImageView. Dalam hal ini, gambar yang digunakan berasal dari drawable resource dengan nama "empty_dice".

Pada baris ke 18, `app:layout_constraintStart_toStartOf="parent"`: Mengikat sisi kiri ImageView ke sisi kiri parent (ConstraintLayout), sehingga ImageView akan berada di sisi kiri layout.

Pada baris ke 19, `app:layout_constraintTop_toTopOf="parent"`: Mengikat sisi atas ImageView ke sisi atas parent (ConstraintLayout), sehingga ImageView akan berada di bagian atas layout. `la<ImageView`: Mendefinisikan sebuah ImageView, yaitu elemen untuk menampilkan gambar atau grafis.

Pada baris ke 22, `android:id="@+id/dice_image2"`: Memberikan identifier (ID) unik untuk ImageView ini agar dapat diidentifikasi dan diakses dalam kode Java atau Kotlin.

Pada baris ke 23, `android:layout_width="150dp"`: Menetapkan lebar ImageView menjadi 150dp (density-independent pixels), sehingga gambar akan memiliki lebar tetap.

Pada baris ke 24, `android:layout_height="150dp"`: Menetapkan tinggi ImageView menjadi 150dp, sehingga gambar akan memiliki tinggi tetap.

Pada baris ke 25, `android:layout_marginTop="312dp"`: Menetapkan margin dari sisi atas ImageView sebesar 312dp.

Pada baris ke 26, `android:layout_marginEnd="36dp"`: Menetapkan margin dari sisi kanan (atau sisi akhir) ImageView sebesar 36dp.

Pada baris ke 27, `android:src="@drawable/empty_dice"`: Menetapkan sumber gambar untuk ditampilkan di ImageView. Dalam hal ini, gambar yang digunakan berasal dari drawable resource dengan nama "empty_dice".

Pada baris ke 28, `app:layout_constraintEnd_toEndOf="parent"`: Mengikat sisi kanan ImageView ke sisi kanan parent (ConstraintLayout), sehingga ImageView akan berada di sisi kanan layout.

Pada baris ke 29, `app:layout_constraintTop_toTopOf="parent"`: Mengikat sisi atas ImageView ke sisi atas parent (ConstraintLayout), sehingga ImageView akan berada di bagian atas layout.

Pada baris ke 31, `<Button`: Mendefinisikan sebuah Button, yaitu elemen untuk menampilkan tombol yang dapat diklik.

Pada baris ke 32, `android:id="@+id/let_roll"`: Memberikan identifier (ID) unik untuk Button ini agar dapat diidentifikasi dan diakses dalam kode Java atau Kotlin.

Pada baris ke 33, `android:layout_width="wrap_content"`: Menetapkan lebar Button agar disesuaikan dengan lebar teks yang ada di dalamnya.

Pada baris ke 34, `android:layout_height="60dp"`: Menetapkan tinggi Button menjadi 60dp (density-independent pixels), sehingga tombol memiliki tinggi tetap.

Pada baris ke 35, `android:layout_marginTop="16dp"`: Menetapkan margin dari sisi atas Button sebesar 16dp.

Pada baris ke 36, `android:backgroundTint="#FF03DAC5"`: Menetapkan warna latar belakang Button. Dalam hal ini, warna yang diberikan adalah #FF03DAC5.

Pada baris ke 37, `android:text="@string/roll_dice"`: Menetapkan teks yang akan ditampilkan pada Button. Teks ini diambil dari resource string dengan nama "roll_dice".

Pada baris ke 38, `android:textColor="#000000"`: Menetapkan warna teks pada Button. Dalam hal ini, warna teks yang diberikan adalah hitam (#000000).

Pada baris ke 39, `app:layout_constraintBaseline_toBottomOf="parent"`: Mengikat baseline (garis dasar) dari Button ke bagian bawah parent (ConstraintLayout), sehingga Button akan terletak pada posisi yang sejajar dengan bagian bawah layout.

Pada baris ke 40, `app:layout_constraintEnd_toEndOf="parent"`: Mengikat sisi kanan Button ke sisi kanan parent (ConstraintLayout), sehingga Button akan berada di sisi kanan layout.

Pada baris ke 41, `app:layout_constraintHorizontal_bias="0.498"`: Menetapkan kecenderungan horizontal Button terhadap parent, dalam hal ini setengah dari lebar parent. Nilai 0.0 berarti ke arah kiri, 1.0 berarti ke arah kanan, dan 0.5 berarti di tengah.

Pada baris ke 44, `app:layout_constraintStart_toStartOf="parent"`: Mengikat sisi kiri Button ke sisi kiri parent (ConstraintLayout), sehingga Button akan berada di sisi kiri layout.

Pada baris ke 45, `app:layout_constraintTop_toBottomOf="@id/dice_image1"`: Mengikat sisi atas Button ke sisi bawah ImageView dengan ID dice_image1, sehingga Button akan berada di bawah ImageView tersebut.

Pada baris ke 47, `</androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah penutup dari elemen ConstraintLayout, yang menandakan akhir dari tata letak XML. Semua elemen yang ditata dalam ConstraintLayout terdapat di antara tag pembuka dan penutup ini.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

[https://github.com/farlyhaydyhdjalil/Praktikum-
PEMROGRAMANMOBILE/tree/774512e4fff5969d626743ab171dc40fd6ee091/Mo
dul%201/Dice%20Role](https://github.com/farlyhaydyhdjalil/Praktikum-PEMROGRAMANMOBILE/tree/774512e4fff5969d626743ab171dc40fd6ee091/Mo
dul%201/Dice%20Role)

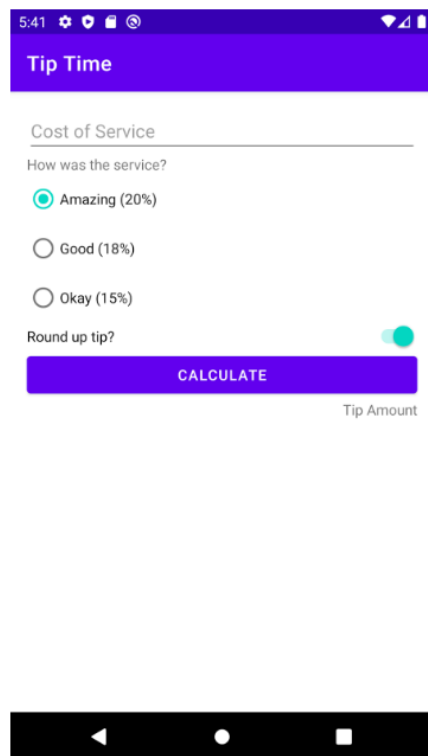
MODUL 2 : ANDROID LAYOUT

SOAL 1

Soal Praktikum:

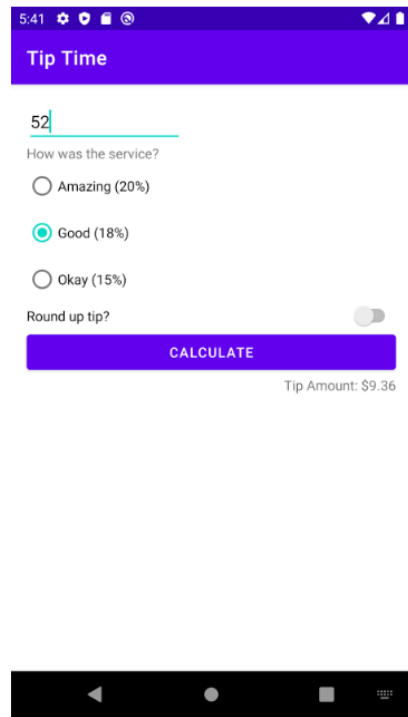
1. Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

1. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
2. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.
3. Pengaturan Pembulatan Tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
4. Tampilan Hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 7. Tampilan Awal Aplikasi

Gambar 1 Tampilan Awal Aplikasi



Gambar 8. Tampilan Aplikasi Setelah Dijalankan

Gambar 2 Tampilan Aplikasi Setelah Dijalankan

A. Source Code

```
1 package com.example.tipcalculatorapp
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import
6     com.example.tipcalculatorapp.databinding.ActivityMainBinding
7 import java.text.NumberFormat
8
9 class MainActivity : AppCompatActivity() {
10     private lateinit var binding:
11         ActivityMainBinding
```

11	
12	override fun onCreate(savedInstanceState:
	Bundle?) {
13	super.onCreate(savedInstanceState)
14	
15	binding =
	ActivityMainBinding.inflate(layoutInflater)
16	setContentView(binding.root)
17	
18	binding.calculateButton.setOnClickListener {
	calculateTip() }
19	}
20	
21	private fun calculateTip() {
22	val stringInTextField =
	binding.costOfService.text.toString()
23	val cost =
	stringInTextField.toDoubleOrNull()
24	if (cost == null) {
25	binding.tipResult.text = ""
26	return
27	}
28	
29	val tipPercentage = when
	(binding.tipOptions.checkedRadioButtonId) {
30	R.id.option_twenty_percent -> 0.20
31	R.id.option_eighteen_percent -> 0.18
32	else -> 0.15
33	}
34	
35	var tip = tipPercentage * cost
36	if (binding.roundUpSwitch.isChecked) {
37	tip = kotlin.math.ceil(tip)
38	}
39	
40	val formattedTip =
	NumberFormat.getCurrencyInstance().format(tip)
41	binding.tipResult.text =
	getString(R.string.tip_amount, formattedTip)
42	}
43	}

Tabel 3. Source Code Soal 1 yang Kotlin

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3
4      xmlns:android="http://schemas.android.com/apk/res/an
5      droid"
6      xmlns:app="http://schemas.android.com/apk/res-
7      auto"
8      xmlns:tools="http://schemas.android.com/tools"
9      android:layout_width="match_parent"
10     android:layout_height="match_parent"
11     android:padding="16dp"
12     tools:context=".MainActivity">
13
14     <EditText
15         android:id="@+id/cost_of_service"
16         android:layout_width="160dp"
17         android:layout_height="wrap_content"
18         android:hint="@string/cost_of_service"
19         android:inputType="numberDecimal"
20
21     app:layout_constraintStart_toStartOf="parent"
22     app:layout_constraintTop_toTopOf="parent" />
23
24     <TextView
25         android:id="@+id/service_question"
26         android:layout_width="wrap_content"
27         android:layout_height="wrap_content"
28         android:text="@string/how_was_the_service"
29
30     app:layout_constraintStart_toStartOf="parent"
31     app:layout_constraintTop_toBottomOf="@id/cost_of_ser
32     vice" />
33
34     <RadioGroup
35         android:id="@+id/tip_options"
36         android:layout_width="wrap_content"
37         android:layout_height="wrap_content"
38
39     android:checkedButton="@id/option_twenty_percent"
40     android:orientation="vertical"

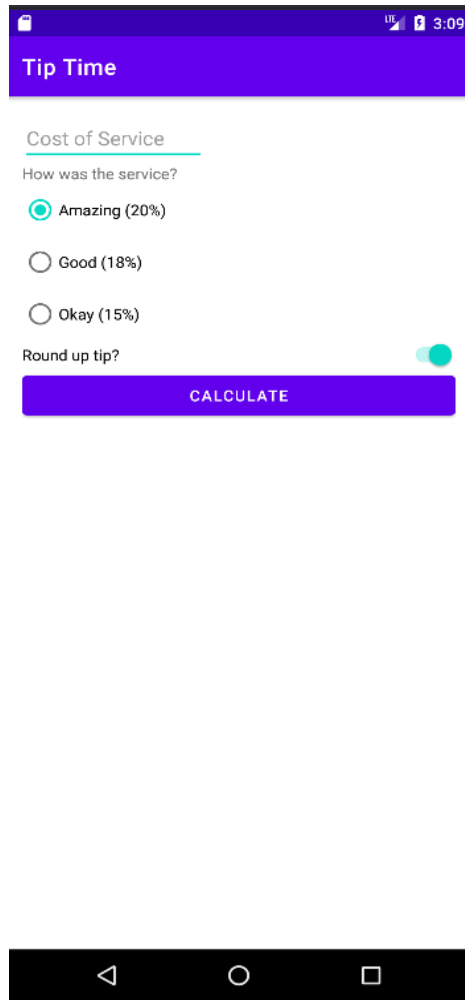
```

34	app:layout_constraintStart_toStartOf="parent"
35	app:layout_constraintTop_toBottomOf="@id/service_question">
36	
37	<RadioButton
38	android:id="@+id/option_twenty_percent"
39	android:layout_width="wrap_content"
40	android:layout_height="wrap_content"
41	android:text="@string/amazing_service"
	/>
42	
43	<RadioButton
44	android:id="@+id/option_eighteen_percent"
45	android:layout_width="wrap_content"
46	android:layout_height="wrap_content"
47	android:text="@string/good_service" />
48	
49	<RadioButton
50	android:id="@+id/option_fifteen_percent"
51	android:layout_width="wrap_content"
52	android:layout_height="wrap_content"
53	android:text="@string/ok_service" />
54	</RadioGroup>
55	
56	<Switch
57	android:id="@+id/round_up_switch"
58	android:layout_width="0dp"
59	android:layout_height="wrap_content"
60	android:checked="true"
61	android:text="@string/round_up_tip"
62	app:layout_constraintEnd_toEndOf="parent"
63	app:layout_constraintStart_toStartOf="@id/tip_options"
64	app:layout_constraintTop_toBottomOf="@id/tip_options"
65	tools:ignore="UseSwitchCompatOrMaterialXml"
	/>
66	

67	<Button
68	android:id="@+id/calculate_button"
69	android:layout_width="0dp"
70	android:layout_height="wrap_content"
71	android:text="@string/calculate"
72	app:layout_constraintEnd_toEndOf="parent"
73	app:layout_constraintStart_toStartOf="parent"
74	app:layout_constraintTop_toBottomOf="@id/round_up_switch" />
75	
76	<TextView
77	android:id="@+id/tip_result"
78	android:layout_width="wrap_content"
79	android:layout_height="wrap_content"
80	tools:text="Tip Amount: \$10"
81	app:layout_constraintEnd_toEndOf="parent"
82	app:layout_constraintTop_toBottomOf="@id/calculate_button" />
83	</androidx.constraintlayout.widget.ConstraintLayout>

Tabel 4. Source Code Soal 1 yang XML

B. Output Program



The screenshot shows a mobile application interface with a blue header bar containing the title "Tip Time". Below the header, there is a section titled "Cost of Service" with a red underline. Underneath, the text "How was the service?" is displayed. There are three radio button options: "Amazing (20%)", "Good (18%)", and "Okay (15%)". The "Amazing (20%)" option is selected, indicated by a red dot. Below these options, there is a toggle switch for "Round up tip?", which is currently turned on (red). At the bottom of the form is a large red button labeled "CALCULATE". The entire application is displayed on a black background, which is a common representation for a mobile device screen in a screenshot.

Gambar 9. Screenshot Hasil Jawaban Soal 1 yang ke 1

The screenshot shows a mobile application interface for calculating a tip. At the top, a purple header bar contains the title "Tip Time". Below the header, a text input field displays the number "52". Underneath the input field, the text "How was the service?" is followed by three radio button options: "Amazing (20%)", "Good (18%)", and "Okay (15%)". The "Good (18%)" option is selected. Below these options is a toggle switch for "Round up tip?". A purple button labeled "CALCULATE" is positioned below the toggle. To the right of the button, the text "Tip Amount: \$9.36" is displayed. The bottom of the screen shows a black navigation bar with three white icons: a back arrow, a circle, and a square.

Gambar 10. Screenshot Hasil Jawaban Soal 1 yang ke 2

C. Pembahasan

Untuk file Kotlin

Pada baris ke 1, package com.example.tipcalculatorapp: Mendeklarasikan paket untuk kelas ini. Semua kelas dalam file ini berada dalam paket com.example.tipcalculatorapp.

Pada baris ke 3, import android.os.Bundle: Mengimpor kelas Bundle dari Android SDK, yang digunakan untuk menyimpan status instance.

Pada baris ke 4, import androidx.appcompat.app.AppCompatActivity: Mengimpor kelas AppCompatActivity dari AndroidX library, yang menyediakan fitur untuk aktivitas yang mendukung backward compatibility.

Pada baris ke 5, import com.example.tipcalculatorapp.databinding.ActivityMainBinding: Mengimpor kelas ActivityMainBinding yang dihasilkan secara otomatis dari file layout XML activity_main.xml. Binding ini memungkinkan akses langsung ke tampilan dalam layout tanpa perlu menggunakan findViewById.

Pada baris ke 6, import java.text.NumberFormat: Mengimpor kelas NumberFormat dari Java untuk memformat angka menjadi bentuk mata uang.

Pada baris ke 8, class MainActivity : AppCompatActivity() {: Mendefinisikan kelas MainActivity, yang merupakan turunan dari AppCompatActivity.

Pada baris ke 10, lateinit var binding: ActivityMainBinding: Mendeklarasikan variabel binding dengan tipe ActivityMainBinding menggunakan lateinit, yang berarti variabel ini akan diinisialisasi nanti sebelum digunakan.

Pada baris ke 12, override fun onCreate(savedInstanceState: Bundle?) {: Mendefinisikan fungsi onCreate yang dipanggil saat aktivitas dibuat. savedInstanceState berisi status yang disimpan dari instance sebelumnya.

Pada baris ke 13, super.onCreate(savedInstanceState): Memanggil metode onCreate dari superclass untuk menjalankan inisialisasi dasar dari AppCompatActivity.

Pada baris ke 15, binding = ActivityMainBinding.inflate(layoutInflater): Menginisialisasi binding dengan memanggil metode inflate pada ActivityMainBinding, yang mengembangkan (inflate) layout dan mengikatnya.

Pada baris ke 16, setContentView(binding.root): Menetapkan tampilan konten dari aktivitas ke root dari binding, yang merupakan layout utama dari activity_main.xml.

Pada baris ke 18, `binding.calculateButton.setOnClickListener { calculateTip() }`: Menetapkan listener `onClick` pada tombol `calculateButton`. Saat tombol diklik, fungsi `calculateTip` akan dipanggil.

Pada baris ke 21, `private fun calculateTip() {`: Mendefinisikan fungsi `calculateTip` yang akan menghitung dan menampilkan tip berdasarkan input pengguna.

Pada baris ke 22, `val stringInTextField = binding.costOfService.text.toString()`: Mengambil teks dari input `costOfService` sebagai string dan menyimpannya dalam variabel `stringInTextField`.

Pada baris ke 23, `val cost = stringInTextField.toDoubleOrNull()`: Mengonversi string `stringInTextField` menjadi `Double` atau `null` jika konversi gagal, dan menyimpannya dalam variabel `cost`.

Pada baris ke 24, `if (cost == null) {`: Memeriksa apakah `cost` bernilai `null`.

Pada baris ke 25, `binding.tipResult.text = ""`: Jika `cost` bernilai `null`, mengatur teks `tipResult` menjadi kosong.

Pada baris ke 26, `return`: Menghentikan eksekusi fungsi `calculateTip` jika `cost` bernilai `null`.

Pada baris ke 29, `val tipPercentage = when (binding.tipOptions.checkedRadioButtonId) { ... }`: Menggunakan ekspresi `when` untuk menentukan persentase tip berdasarkan ID dari radio button yang dipilih dalam `tipOptions`.

Pada baris ke 30, `R.id.option_twenty_percent -> 0.20`: Jika radio button dengan ID `option_twenty_percent` dipilih, maka `tipPercentage` adalah 0.20 (20%).

Pada baris ke 31, `R.id.option_eighteen_percent -> 0.18`: Jika radio button dengan ID `option_eighteen_percent` dipilih, maka `tipPercentage` adalah 0.18 (18%).

Pada baris ke 32, `else -> 0.15`: Jika radio button lain dipilih, `tipPercentage` adalah 0.15 (15%).

Pada baris ke 35, `var tip = tipPercentage * cost`: Menghitung tip dengan mengalikan `tipPercentage` dengan `cost`, dan menyimpannya dalam variabel `tip`.

Pada baris ke 36, `if (binding.roundUpSwitch.isChecked) {`: Memeriksa apakah switch `roundUpSwitch` dinyalakan.

Pada baris ke 37, `tip = kotlin.math.ceil(tip)`: Jika switch dinyalakan, maka nilai `tip` dibulatkan ke atas menggunakan `ceil`.

Pada baris ke 40, `val formattedTip = NumberFormat.getCurrencyInstance().format(tip)`: Memformat nilai tip menjadi bentuk mata uang yang sesuai dengan lokal saat ini, dan menyimpannya dalam variabel `formattedTip`.

Pada baris ke 41, `binding.tipResult.text = getString(R.string.tip_amount, formattedTip)`: Mengatur teks `tipResult` untuk menampilkan string yang diformat dengan nilai `formattedTip`. `getString(R.string.tip_amount, formattedTip)` mengambil string dari sumber daya dengan placeholder yang diisi oleh `formattedTip`.

Pada baris ke 42, `}`: Menutup fungsi `calculateTip`.

Pada baris ke 43, `}`: Menutup kelas `MainActivity`.

Untuk file XML

Pada baris ke 1, `<?xml version="1.0" encoding="utf-8"?>`: Ini adalah deklarasi XML yang menunjukkan versi XML yang digunakan (1.0) dan encoding yang digunakan (utf-8).

Pada baris ke 2, `<androidx.constraintlayout.widget.ConstraintLayout ...>`: Ini adalah tag pembuka untuk layout yang menggunakan `ConstraintLayout`, tata letak yang fleksibel untuk desain antarmuka pengguna di Android.

Pada baris ke 3, `xmlns:android="http://schemas.android.com/apk/res/android"`: Ini adalah namespace yang digunakan untuk elemen Android dalam file XML.

Pada baris ke 4, `xmlns:app="http://schemas.android.com/apk/res-auto"`: Ini adalah namespace yang digunakan untuk elemen dari library support (support library) dalam file XML.

Pada baris ke 5, `xmlns:tools="http://schemas.android.com/tools"`: Ini adalah namespace yang digunakan untuk atribut yang hanya dipakai saat desain atau pembangunan (build) (misalnya, untuk preview layout di Android Studio).

Pada baris ke 6 dan 7, `android:layout_width="match_parent"` dan `android:layout_height="match_parent"`: Ini menetapkan lebar dan tinggi dari layout menjadi seukuran parent-nya, sehingga layout akan mengisi seluruh ruang yang tersedia.

Pada baris ke 8, `android:padding="16dp"`: Ini menetapkan jarak padding sebesar 16dp dari sisi layout, yang akan memberikan ruang tambahan di sekeliling konten dalam layout.

Pada baris ke 9, `tools:context=".MainActivity"`: Ini adalah atribut yang digunakan untuk menunjukkan kelas Activity yang terkait dengan layout ini, digunakan terutama untuk tujuan preview di Android Studio.

Pada baris ke 11, `<EditText ... />`: Ini adalah widget EditText yang memungkinkan pengguna untuk memasukkan teks. Atribut-atribut seperti id, lebar, tinggi, hint, dan tipe input ditentukan di sini.

Pada baris ke 12, `android:id="@+id/cost_of_service"`: Ini adalah atribut yang memberikan ID unik untuk EditText. ID ini akan digunakan untuk mengidentifikasi EditText dalam kode Java/Kotlin untuk mengakses dan memanipulasi isinya.

Pada baris ke 13, `android:layout_width="160dp"`: Ini adalah atribut yang menetapkan lebar EditText menjadi 160dp. Dengan demikian, lebar EditText akan tetap konstan, tidak bergantung pada konten di dalamnya.

Pada baris ke 14, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi EditText agar disesuaikan dengan konten di dalamnya. Tinggi EditText akan menyesuaikan dengan tinggi teks atau konten yang dimasukkan pengguna.

Pada baris ke 15, `android:hint="@string/cost_of_service"`: Ini adalah atribut yang menetapkan teks petunjuk (hint) yang muncul di dalam EditText saat tidak ada teks yang dimasukkan. Nilai dari atribut ini diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 16, `android:inputType="numberDecimal"`: Ini adalah atribut yang menetapkan jenis input yang diperbolehkan untuk EditText. Dalam hal ini, EditText hanya akan menerima input berupa angka desimal.

Pada baris ke 17, `app:layout_constraintStart_toStartOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi mulai (start) EditText terhadap parent layout. Dalam hal ini, EditText akan diatur mulai dari sisi kiri (start) dari parent layout.

Pada baris ke 18, `app:layout_constraintTop_toTopOf="parent"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) EditText terhadap parent layout. Dalam hal ini, EditText akan diatur mulai dari sisi atas (top) dari parent layout.

Pada baris ke 20, `<TextView ... />`: Ini adalah widget TextView untuk menampilkan teks. Di sini, teks yang ditampilkan adalah pertanyaan tentang kualitas layanan.

Pada baris ke 21, `android:id="@+id/service_question"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini akan digunakan untuk mengidentifikasi TextView dalam kode Java/Kotlin untuk mengakses dan memanipulasi teksnya.

Pada baris ke 22, `android:layout_width="wrap_content"`: Ini adalah atribut yang menetapkan lebar TextView agar disesuaikan dengan konten teks di dalamnya. Lebar TextView akan menyesuaikan dengan lebar teks yang ditampilkan.

Pada baris ke 23, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi TextView agar disesuaikan dengan konten teks di dalamnya. Tinggi TextView akan menyesuaikan dengan tinggi teks yang ditampilkan.

Pada baris ke 24, `android:text="@string/how_was_the_service"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di dalam TextView. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 25, `app:layout_constraintStart_toStartOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi mulai (start) TextView terhadap parent layout. Dalam hal ini, TextView akan diatur mulai dari sisi kiri (start) dari parent layout.

Pada baris ke 26, `app:layout_constraintTop_toBottomOf="@id/cost_of_service"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) TextView terhadap elemen lain di dalam layout. Dalam hal ini, TextView akan diatur di bawah (below) elemen dengan ID `cost_of_service`.

Pada baris ke 28 dan 54, `<RadioGroup ...>` dan `<RadioButton ... />`: Ini adalah grup radio button untuk memilih opsi tip berdasarkan kualitas layanan. RadioButton diatur dalam RadioGroup.

Pada baris ke 29, `android:id="@+id/tip_options"`: Ini adalah atribut yang memberikan ID unik untuk RadioGroup. ID ini akan digunakan untuk mengidentifikasi RadioGroup dalam kode Java/Kotlin untuk mengakses dan memanipulasi pilihan pengguna.

Pada baris ke 30, `android:layout_width="wrap_content"`: Ini adalah atribut yang menetapkan lebar RadioGroup agar disesuaikan dengan konten di dalamnya. Lebar RadioGroup akan menyesuaikan dengan lebar pilihan radio di dalamnya.

Pada baris ke 31, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi RadioGroup agar disesuaikan dengan konten di dalamnya. Tinggi RadioGroup akan menyesuaikan dengan tinggi pilihan radio di dalamnya.

Pada baris ke 32, `android:checkedButton="@id/option_twenty_percent"`: Ini adalah atribut yang menetapkan pilihan radio yang akan diperiksa secara default ketika RadioGroup ditampilkan. Dalam hal ini, pilihan radio dengan ID `option_twenty_percent` akan diperiksa secara default.

Pada baris ke 33, `android:orientation="vertical"`: Ini adalah atribut yang menentukan orientasi dari `RadioGroup`. Dalam hal ini, orientasinya diatur menjadi vertikal, yang berarti pilihan radio akan ditumpuk satu per satu secara vertikal.

Pada baris ke 34, `app:layout_constraintStart_toStartOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi mulai (start) `RadioGroup` terhadap parent layout. Dalam hal ini, `RadioGroup` akan diatur mulai dari sisi kiri (start) dari parent layout.

Pada baris ke 35, `app:layout_constraintTop_toBottomOf="@id/service_question"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) `RadioGroup` terhadap elemen lain di dalam layout. Dalam hal ini, `RadioGroup` akan diatur di bawah (below) elemen dengan ID `service_question`.

Pada baris ke 38, `android:id="@+id/option_twenty_percent"`: Ini adalah atribut yang memberikan ID unik untuk `RadioButton`. ID ini akan digunakan untuk mengidentifikasi `RadioButton` dalam kode Java/Kotlin untuk mengakses dan memanipulasi pilihannya.

Pada baris ke 39, `android:layout_width="wrap_content"`: Ini adalah atribut yang menetapkan lebar `RadioButton` agar disesuaikan dengan konten teks di dalamnya. Lebar `RadioButton` akan menyesuaikan dengan lebar teks yang ditampilkan.

Pada baris ke 40, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi `RadioButton` agar disesuaikan dengan konten teks di dalamnya. Tinggi `RadioButton` akan menyesuaikan dengan tinggi teks yang ditampilkan.

Pada baris ke 41, `android:text="@string/amazing_service"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di dalam `RadioButton`. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 44, `android:id="@+id/option_eighteen_percent"`: Ini adalah atribut yang memberikan ID unik untuk `RadioButton`. ID ini akan digunakan untuk mengidentifikasi `RadioButton` dalam kode Java/Kotlin untuk mengakses dan memanipulasi pilihannya.

Pada baris ke 45, `android:layout_width="wrap_content"`: Atribut ini menentukan lebar `RadioButton` agar sesuai dengan konten teks di dalamnya. Lebar `RadioButton` akan menyesuaikan dengan lebar teks yang ditampilkan.

Pada baris ke 46, `android:layout_height="wrap_content"`: Atribut ini menentukan tinggi `RadioButton` agar sesuai dengan konten teks di dalamnya. Tinggi `RadioButton` akan menyesuaikan dengan tinggi teks yang ditampilkan.

Pada baris ke 47, `android:text="@string/good_service"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di dalam RadioButton. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 50, `android:id="@+id/option_fifteen_percent"`: Ini adalah atribut yang memberikan ID unik untuk RadioButton. ID ini akan digunakan untuk mengidentifikasi RadioButton dalam kode Java/Kotlin untuk mengakses dan memanipulasi pilihannya.

Pada baris ke 51, `android:layout_width="wrap_content"`: Atribut ini menentukan lebar RadioButton agar sesuai dengan konten teks di dalamnya. Lebar RadioButton akan menyesuaikan dengan lebar teks yang ditampilkan.

Pada baris ke 52, `android:layout_height="wrap_content"`: Atribut ini menentukan tinggi RadioButton agar sesuai dengan konten teks di dalamnya. Tinggi RadioButton akan menyesuaikan dengan tinggi teks yang ditampilkan.

Pada baris ke 53, `android:text="@string/ok_service"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di dalam RadioButton. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 54, `</RadioGroup>`: Ini adalah tag penutup untuk elemen RadioGroup. Ini menandakan akhir dari RadioGroup dan menutup lingkup semua RadioButton yang ditambahkan ke dalamnya. Semua RadioButton yang ditempatkan sebelumnya dalam kode XML berada di dalam RadioGroup ini.

Pada baris ke 56, `<Switch ... />`: Ini adalah switch (sakelar) yang memungkinkan pengguna untuk memilih apakah tip akan dibulatkan ke atas atau tidak.

Pada baris ke 57, `android:id="@+id/round_up_switch"`: Ini adalah atribut yang memberikan ID unik untuk Switch. ID ini akan digunakan untuk mengidentifikasi Switch dalam kode Java/Kotlin untuk mengakses dan memanipulasi statusnya.

Pada baris ke 58, `android:layout_width="0dp"`: Ini adalah atribut yang menetapkan lebar Switch menjadi nol dp. Pengaturan lebar ini diperlukan ketika menggunakan ConstraintLayout dan ingin mengontrol lebar elemen menggunakan constraint.

Pada baris ke 59, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi Switch agar disesuaikan dengan konten di dalamnya. Tinggi Switch akan menyesuaikan dengan tinggi konten teks atau tombol di dalamnya.

Pada baris ke 60, `android:checked="true"`: Ini adalah atribut yang menentukan apakah Switch dalam keadaan aktif atau tidak saat pertama kali ditampilkan. Dalam hal ini, Switch akan ditampilkan dalam keadaan aktif.

Pada baris ke 61, `android:text="@string/round_up_tip"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di samping Switch. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 62, `app:layout_constraintEnd_toEndOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi akhir (end) Switch terhadap parent layout. Dalam hal ini, Switch akan diatur di sisi kanan (end) dari parent layout.

Pada baris ke 63, `app:layout_constraintStart_toStartOf="@id/tip_options"`: Ini adalah atribut yang menetapkan constraint posisi mulai (start) Switch terhadap elemen lain di dalam layout. Dalam hal ini, Switch akan diatur mulai dari sisi kiri (start) dari elemen dengan ID `tip_options`.

Pada baris ke 64, `app:layout_constraintTop_toBottomOf="@id/tip_options"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) Switch terhadap elemen lain di dalam layout. Dalam hal ini, Switch akan diatur di bawah (below) elemen dengan ID `tip_options`.

Pada baris ke 65, `tools:ignore="UseSwitchCompatOrMaterialXml"`: Ini adalah atribut yang digunakan untuk mengabaikan peringatan dari alat pengembangan. Dalam hal ini, peringatan tersebut berkaitan dengan penggunaan elemen Switch, yang direkomendasikan untuk diganti dengan elemen `SwitchCompat` atau elemen `Material Components`.

Pada baris ke 67, `<Button ... />`: Ini adalah tombol yang akan digunakan untuk menghitung tip berdasarkan input pengguna.

Pada baris ke 68, `android:id="@+id/calculate_button"`: Ini adalah atribut yang memberikan ID unik untuk Button. ID ini akan digunakan untuk mengidentifikasi Button dalam kode Java/Kotlin untuk menambahkan fungsi event `onClick()` atau melakukan manipulasi lainnya.

Pada baris ke 69, `android:layout_width="0dp"`: Ini adalah atribut yang menetapkan lebar Button menjadi 0dp. Pengaturan lebar ini diperlukan ketika menggunakan `ConstraintLayout` dan ingin mengontrol lebar elemen menggunakan constraint.

Pada baris ke 70, `android:layout_height="wrap_content"`: Ini adalah atribut yang menetapkan tinggi Button agar disesuaikan dengan konten teks di dalamnya. Tinggi Button akan menyesuaikan dengan tinggi konten teks yang ditampilkan.

Pada baris ke 71, `android:text="@string/calculate"`: Ini adalah atribut yang menetapkan teks yang akan ditampilkan di dalam Button. Nilai teks diambil dari sumber daya string yang didefinisikan di dalam file `strings.xml`.

Pada baris ke 72, `app:layout_constraintEnd_toEndOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi akhir (end) Button terhadap parent layout. Dalam hal ini, Button akan diatur di sisi kanan (end) dari parent layout.

Pada baris ke 73, `app:layout_constraintStart_toStartOf="parent"`: Ini adalah atribut yang menetapkan constraint posisi mulai (start) Button terhadap parent layout. Dalam hal ini, Button akan diatur mulai dari sisi kiri (start) dari parent layout.

Pada baris ke 74, `app:layout_constraintTop_toBottomOf="@id/round_up_switch"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) Button terhadap elemen lain di dalam layout. Dalam hal ini, Button akan diatur di bawah (below) elemen dengan ID `round_up_switch`.

Pada baris ke 76, `<TextView ... />`: Ini adalah TextView untuk menampilkan hasil perhitungan tip.

Pada baris ke 77, `android:id="@+id/tip_result"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini akan digunakan untuk mengidentifikasi TextView dalam kode Java/Kotlin untuk mengakses dan memanipulasi teksnya.

Pada baris ke 78, `android:layout_width="wrap_content"`: Atribut ini menentukan lebar TextView agar sesuai dengan konten teks di dalamnya. Lebar TextView akan menyesuaikan dengan lebar teks yang ditampilkan.

Pada baris ke 79, `android:layout_height="wrap_content"`: Atribut ini menentukan tinggi TextView agar sesuai dengan konten teks di dalamnya. Tinggi TextView akan menyesuaikan dengan tinggi teks yang ditampilkan.

Pada baris ke 80, `tools:text="Tip Amount: $10"`: Ini adalah atribut alat pengembangan yang menetapkan teks yang akan ditampilkan di dalam TextView saat digunakan di lingkungan pengembangan (preview). Ini tidak akan ditampilkan saat aplikasi dijalankan di perangkat.

Pada baris ke 81, `app:layout_constraintEnd_toEndOf="parent"`: Ini adalah atribut yang menetapkan constraint (batasan) posisi akhir (end) TextView terhadap parent layout. Dalam hal ini, TextView akan diatur di sisi kanan (end) dari parent layout.

Pada baris ke 82, `app:layout_constraintTop_toBottomOf="@id/calculate_button"`: Ini adalah atribut yang menetapkan constraint posisi atas (top) TextView terhadap elemen lain di dalam layout. Dalam hal ini, TextView akan diatur di bawah (below) elemen dengan ID `calculate_button`.

Pada baris ke 83, `</androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah tag penutup untuk elemen `ConstraintLayout`. Ini menandakan akhir dari `ConstraintLayout` dan menutup lingkup semua elemen yang ditambahkan ke dalamnya.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/farlyhaydyhdjalil/Praktikum-PEMROGRAMANMOBILE/tree/e93522abdeece3e91dc975485016e7f50afcd4d/Modul%202/Calculator%20Tip>

MODUL 3 : ANDROID NAVIGATION

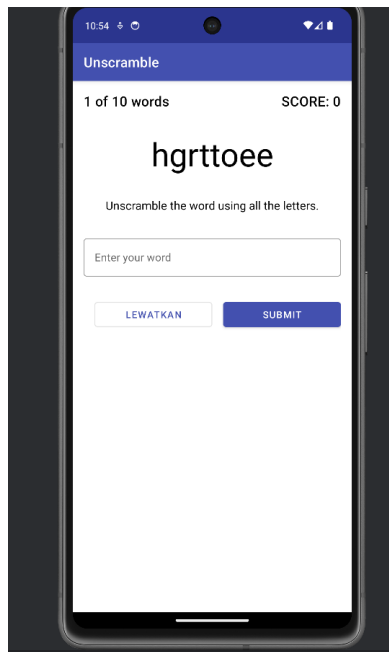
SOAL 1

Soal Praktikum:

1. Buat sebuah aplikasi Android sederhana yang disebut "Unscramble" yang tampilannya menggunakan bahasa Indonesia. Aplikasi ini akan menampilkan kata-kata yang diacak sebanyak 10 soal, dan pengguna harus menebak kata yang benar. Soal yang tidak bisa dijawab bisa dilewati dan setiap satu soal yang benar bernilai 20.

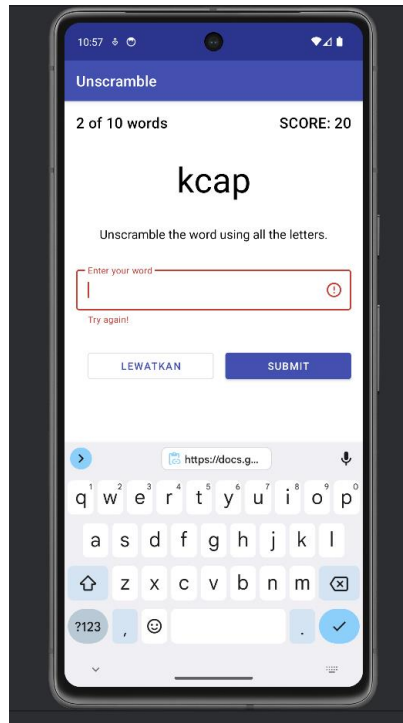
Contoh Hasil Aplikasi :

1. Saat menjawab quiz



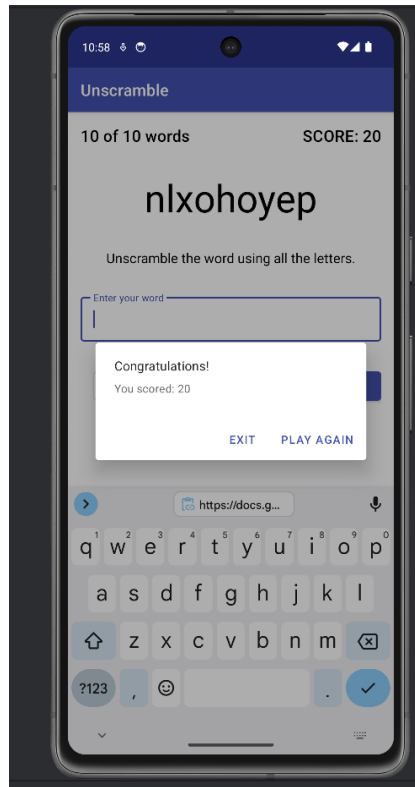
Gambar 11. Contoh pertama

2. Saat field input dibiarkan kosong dan disubmit



Gambar 12. Contoh kedua

3. Menampilkan hasil score akhir setelah 10 soal terjawab



Gambar 13. Contoh Ketiga

A. Source Code

GameFragment.kt

```

1 package com.example.android.unscramble.ui.game
2
3 import android.os.Bundle
4 import android.view.LayoutInflater
5 import android.view.View
6 import android.view.ViewGroup
7 import androidx.databinding.DataBindingUtil
8 import androidx.fragment.app.Fragment
9 import androidx.fragment.app.viewModels
10 import com.example.android.unscramble.R
11 import
12     com.example.android.unscramble.databinding.GameFragm
entBinding
12 import
    com.google.android.material.dialog.MaterialAlertDial

```

```

ogBuilder
13
14 class GameFragment : Fragment() {
15
16     private lateinit var binding:
GameFragmentBinding
17
18     private val viewModel: GameViewModel by
viewModels()
19
20     override fun onCreateView(
21         inflater: LayoutInflater, container:
ViewGroup?,
22         savedInstanceState: Bundle?
23     ): View {
24
25         binding = DataBindingUtil.inflate(inflater,
R.layout.game_fragment, container, false)
26         return binding.root
27     }
28
29     override fun onViewCreated(view: View,
savedInstanceState: Bundle?) {
30         super.onViewCreated(view,
savedInstanceState)
31
32         binding.gameViewModel = viewModel
33         binding.maxNoOfWords = MAX_NO_OF_WORDS
34
35         binding.lifecycleOwner = viewLifecycleOwner
36
37         binding.submit.setOnClickListener {
onSubmitWord() }
38         binding.skip.setOnClickListener {
onSkipWord() }
39     }
40
41     private fun onSubmitWord() {
42         val playerWord =
binding.textInputEditText.text.toString()
43
44         if (viewModel.isUserWordCorrect(playerWord))
{

```



```

45         setErrorTextField(false)
46         if (!viewModel.nextWord()) {
47             showFinalScoreDialog()
48         }
49     } else {
50         setErrorTextField(true)
51     }
52 }
53
54 private fun onSkipWord() {
55     if (viewModel.nextWord()) {
56         setErrorTextField(false)
57     } else {
58         showFinalScoreDialog()
59     }
60 }
61
62 private fun showFinalScoreDialog() {
63     MaterialAlertDialogBuilder(requireContext())
64         .setTitle(getString(R.string.congratulations))
65         .setMessage(getString(R.string.you_scored,
66             viewModel.score.value))
67         .setCancelable(false)
68         .setNegativeButton(getString(R.string.exit)) { _, _
69             ->
70                 exitGame()
71             }
72         .setPositiveButton(getString(R.string.play_again)) {
73             _, _ ->
74                 restartGame()
75             }
76         .show()
77     }
78
79     private fun restartGame() {
80         viewModel.reinitializeData()
81         setErrorTextField(false)
82     }

```

```

81     private fun exitGame() {
82         activity?.finish()
83     }
84
85     private fun setErrorTextField(error: Boolean) {
86         if (error) {
87             binding.textField.isErrorEnabled = true
88             binding.textField.error =
89 getString(R.string.try_again)
90         } else {
91             binding.textField.isErrorEnabled = false
92             binding.textInputEditText.text = null
93         }
94     }

```

Tabel 5. Source Code Soal 1 Kotlin yang pertama

GameViewModel.kt

```

1  package com.example.android.unscramble.ui.game
2
3  import android.text.Spannable
4  import android.text.SpannableString
5  import android.text.style.TtsSpan
6  import android.util.Log
7  import androidx.lifecycle.LiveData
8  import androidx.lifecycle.MutableLiveData
9  import androidx.lifecycle.Transformations
10 import androidx.lifecycle.ViewModel
11
12 class GameViewModel : ViewModel() {
13     private val _score = MutableLiveData(0)
14     val score: LiveData<Int>
15         get() = _score
16
17     private val _currentWordCount =
18 MutableLiveData(0)
19     val currentWordCount: LiveData<Int>
20         get() = _currentWordCount
21
22     private val _currentScrambledWord =
23 MutableLiveData<String>()
24     val currentScrambledWord: LiveData<Spannable> =
25 Transformations.map(_currentScrambledWord) {

```

```

23         if (it == null) {
24             SpannableString("")
25         } else {
26             val scrambledWord = it.toString()
27             val spannable: Spannable =
28                 SpannableString(scrambledWord)
29                 spannable.setSpan(
30                     TtsSpan.VerbatimBuilder(scrambledWord).build(),
31                     0,
32                     scrambledWord.length,
33                     Spannable.SPAN_INCLUSIVE_INCLUSIVE
34                 )
35             spannable
36         }
37     private var wordsList: MutableList<String> =
38     mutableListOf()
39     private lateinit var currentWord: String
40     init {
41         getNextWord()
42     }
43     private fun getNextWord() {
44         currentWord = allWordsList.random()
45         val tempWord = currentWord.toCharArray()
46         tempWord.shuffle()
47         while (String(tempWord).equals(currentWord,
48 false)) {
49             tempWord.shuffle()
50         }
51         if (wordsList.contains(currentWord)) {
52             getNextWord()
53         } else {
54             Log.d("Unscramble", "currentWord=
55 $currentWord")
56             _currentScrambledWord.value =
57 String(tempWord)
58             _currentWordCount.value =

```

58	<code>_currentWordCount.value?.inc()</code>
	<code>wordsList.add(currentWord)</code>
59	<code>}</code>
60	
61	<code>fun reinitializeData() {</code>
62	<code> _score.value = 0</code>
63	<code> _currentWordCount.value = 0</code>
64	<code> wordsList.clear()</code>
65	<code> getNextWord()</code>
66	<code>}</code>
67	
68	<code> private fun increaseScore() {</code>
69	<code> _score.value =</code>
70	<code>_score.value?.plus(SCORE_INCREASE)</code>
71	<code> }</code>
72	
72	<code> fun isUserWordCorrect(playerWord: String):</code>
73	<code>Boolean {</code>
74	<code> if (playerWord.equals(currentWord, true)) {</code>
	<code> increaseScore()</code>
75	<code> return true</code>
	<code> }</code>
76	<code> return false</code>
77	<code> }</code>
78	
79	<code> fun nextWord(): Boolean {</code>
80	<code> return if (_currentWordCount.value!! <</code>
81	<code>MAX_NO_OF_WORDS) {</code>
82	<code> getNextWord()</code>
83	<code> true</code>
	<code> } else false</code>
84	<code> }</code>
85	<code>}</code>
86	
87	
88	

Tabel 6. Source Code Soal 1 Kotlin yang kedua

ListofWords.kt

1	<code>package com.example.android.unscramble.ui.game</code>
2	
3	

```

4  const val MAX_NO_OF_WORDS = 10
5  const val SCORE_INCREASE = 20
6
7  // List with all the words for the Game
8  val allWordsList: List<String> =
9      listOf("apel", "angin", "anak", "asap", "asri",
10         "awan", "ayam", "ajar", "amal", "adab",
11         "air", "aku", "ayam", "api", "aduh",
12         "anjing", "antar", "arah", "abadi", "alun",
13         "buku", "bola", "ban", "batu", "baik",
14         "buah", "baju", "bangku", "bunga", "belajar",
15         "bambu", "besar", "belut", "berita",
16         "burung", "cinta", "cawan", "cari", "cahaya", "cat",
17         "cuka", "cacing", "ceri", "celana", "cerah",
18         "celana", "cabang", "cetak", "cerdas", "cukur",
19         "daun", "dasi", "daging", "dapur", "dada",
20         "dekat", "diri", "domba", "duduk", "darat",
21         "durian", "dalam", "damai", "danau",
22         "dosen", "elang", "empat", "es", "emas", "enak",
23         "empat", "elok", "ekor", "epal", "embun",
24         "fajar", "film", "foto", "fasilitas", "faktor",
25         "fiksi", "final", "firasat", "fantasi",
26         "flora", "furnitur", "firman", "fisika", "formal",
27         "fitur",
28         "gelas", "garam", "gula", "gajah", "gunung",
29         "gembira", "guntur", "gabus", "gadis", "gajah",
30         "gagang", "gantung", "gantung", "gas",
31         "gerak", "hujan", "hutan", "hari", "hati", "hewan",
32         "hidup", "harum", "hotel", "hitam",
33         "halaman", "ikan", "indah", "ilmu", "istana",
34         "inspirasi",
35         "jalan", "jendela", "jam", "jatuh", "jari",
36         "jambu", "jaring", "jernih", "jeruk", "jawaban",
37         "kucing", "kuda", "kursi", "kertas",
38         "kapal", "kota", "kaki", "kamar", "kasur",
39         "kayak", "langit", "laut", "lilin",
40         "lembut", "lentera", "lintas", "layar", "lebar",
41         "lembah", "lampu",
42         "matahari", "meja", "makan", "minum",
43         "manis", "mimpi", "mobil", "muda", "mudah", "musik",
44         "nasi", "nyamuk", "nangka", "naik", "nyata",
45         "nyanyi", "nona", "novel", "namun", "nara",
46         "ombak", "oren", "obyek", "oleh", "otak",

```

27	"otot", "ongkos", "opini", "olah", "obat", "pohon", "pisau", "pantai", "padi", "pagar", "payung", "piring", "pulau", "pusat", "panas",
28	"papan", "pintu", "pedas", "pelangi", "pena", "quran", "kuadrat", "quark", "quasar", "quiz",
29	"robot", "rumah", "rusa", "roda", "rapi", "rak", "rawat", "rasa", "rakit", "rambut",
30	"sawah", "sapi", "sinar", "susu", "sawah", "sepeda", "surat", "sore", "sopan", "sarung",
31	"taman", "tanah", "tanda", "tangan", "tali", "tahu", "takut", "tas", "tungku", "topi",
32	"ular", "udara", "ubi", "ulang", "umur", "upah", "udang", "usaha", "utama", "uang",
33	"vaksin", "vila", "vokal", "valuta", "vital", "venus", "varian", "virus", "vitamin", "volume",
34	"waktu", "warna", "wajah", "waktu", "wangi", "warung", "wadah", "wawasan", "wacana", "waspada",
35	"xilofon", "xenon", "xilografi", "xerofit", "xenofobia", "yoghurt", "yak", "yodium", "yoyo", "yoga",
36	"yakini", "yuk", "yatim", "yul", "yumpie", "zebra", "zaman", "zakat", "zodiak", "zona")

Tabel 7. Source Code Soal 1 Kotlin yang ketiga

MainActivity.kt

1	package com.example.android.unscramble
2	
3	import android.os.Bundle
4	import androidx.appcompat.app.AppCompatActivity
5	
6	/**
7	* Creates an Activity that hosts the Game fragment
8	in the app
9	*/
10	class MainActivity : AppCompatActivity() {
11	override fun onCreate(savedInstanceState:
12	Bundle?) {
	super.onCreate(savedInstanceState)

13	setContentView(R.layout.main_activity)
14	}
15	}

Tabel 8. Source Code Soal 1 Kotlin yang keempat

game_fragment.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	
3	<layout
	xmlns:android="http://schemas.android.com/apk/res/a
	ndroid"
4	xmlns:app="http://schemas.android.com/apk/res-
	auto"
5	xmlns:tools="http://schemas.android.com/tools">
6	
7	<data>
8	<variable
9	name="gameViewModel"
10	type="com.example.android.unscramble.ui.game.GameVi
	ewModel" />
11	<variable
12	name="maxNoOfWords"
13	type="int" />
14	</data>
15	
16	<ScrollView
17	android:layout_width="match_parent"
18	android:layout_height="match_parent">
19	
20	<androidx.constraintlayout.widget.ConstraintLayout
21	android:layout_width="match_parent"
22	android:layout_height="wrap_content"
23	android:padding="@dimen/default_padding"
24	tools:context=".ui.game.GameFragment">
25	
26	<Button
27	android:id="@+id/skip"
28	style="?attr/materialButtonOutlinedStyle"
29	android:layout_width="0dp"

30	android:layout_height="wrap_content"
31	android:layout_marginStart="@dimen/default_padding"
32	android:layout_marginEnd="@dimen/default_padding"
33	android:text="@string/skip"
34	app:layout_constraintBaseline_toBaselineOf="@+id/submit"
35	app:layout_constraintEnd_toStartOf="@+id/submit"
36	app:layout_constraintStart_toStartOf="parent" />
37	
38	<Button
39	android:id="@+id/submit"
40	android:layout_width="0dp"
41	android:layout_height="wrap_content"
42	android:layout_marginTop="@dimen/default_margin"
43	android:text="@string/submit"
44	app:layout_constraintEnd_toEndOf="parent"
45	app:layout_constraintStart_toEndOf="@+id/skip"
46	app:layout_constraintTop_toBottomOf="@+id/textField"
47	" />
48	<TextView
49	android:id="@+id/textView_instructions"
50	android:layout_width="wrap_content"
51	android:layout_height="wrap_content"
52	android:text="@string/instructions"
53	android:textSize="17sp"
54	app:layout_constraintBottom_toTopOf="@+id/textField"
	"

55	app:layout_constraintEnd_toEndOf="parent"
56	app:layout_constraintStart_toStartOf="parent"
57	app:layout_constraintTop_toBottomOf="@+id/textView_unscrambled_word" />
58	
59	<TextView
60	android:id="@+id/textView_unscrambled_word"
61	android:layout_width="wrap_content"
62	android:layout_height="wrap_content"
63	android:layout_marginTop="@dimen/default_margin"
64	android:layout_marginBottom="@dimen/default_margin"
65	android:text="@{gameViewModel.currentScrambledWord}"
66	android:textAppearance="@style/TextAppearance.MaterialComponents.Headline3"
67	app:layout_constraintBottom_toTopOf="@+id/textView_instructions"
68	app:layout_constraintEnd_toEndOf="parent"
69	app:layout_constraintStart_toStartOf="parent"
70	app:layout_constraintTop_toBottomOf="@+id/word_count"
71	tools:text="Scramble word" />
72	
73	<TextView
74	android:id="@+id/word_count"
75	android:layout_width="wrap_content"
76	android:layout_height="wrap_content"
77	android:text="@{@string/word_count(gameViewModel.currentWordCount, maxNoOfWords)}"

78	android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"
79	app:layout_constraintBottom_toTopOf="@+id/textView_unscrambled_word"
80	app:layout_constraintStart_toStartOf="parent"
81	app:layout_constraintTop_toTopOf="parent"
82	tools:text="3 of 10 words" />
83	
84	<TextView
85	android:id="@+id/score"
86	android:layout_width="wrap_content"
87	android:layout_height="wrap_content"
88	android:text="@{getString(score(gameViewModel.score))}"
89	android:textAllCaps="true"
90	android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"
91	app:layout_constraintEnd_toEndOf="parent"
92	app:layout_constraintTop_toTopOf="parent"
93	tools:text="Score: 20" />
94	
95	<com.google.android.material.textfield.TextInputLayout
96	out
97	android:id="@+id/textField"
98	style="@style/Widget.Unscramble.TextInputLayout.OutlinedBox"
99	android:layout_width="0dp"
100	android:layout_height="wrap_content"
	android:layout_marginTop="@dimen/default_margin"

101	android:hint="@string/enter_your_word"
102	app:errorIconDrawable="@drawable/ic_error"
103	app:helperTextTextAppearance="@style/TextAppearance MaterialComponents.Subtitle1"
104	app:layout_constraintBottom_toTopOf="@+id/submit"
105	app:layout_constraintEnd_toEndOf="parent"
106	app:layout_constraintStart_toStartOf="parent"
107	app:layout_constraintTop_toBottomOf="@+id/textView_ instructions">
108	
109	<com.google.android.material.textfield.TextInputEdi tText
110	android:id="@+id/text_input_edit_text"
111	android:layout_width="match_parent"
112	android:layout_height="match_parent"
113	android:inputType="textPersonName textNoSuggestions "
114	android:maxLines="1" />
115	</com.google.android.material.textfield.TextInputLa yout>
116	
117	</androidx.constraintlayout.widget.ConstraintLayout >
118	</ScrollView>
119	</layout>

Tabel 9. Source Code Soal 1 XML yang pertama

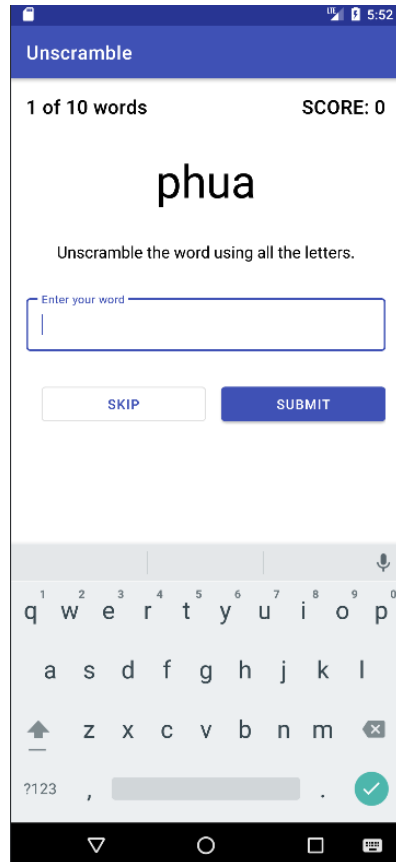
main_activity.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout

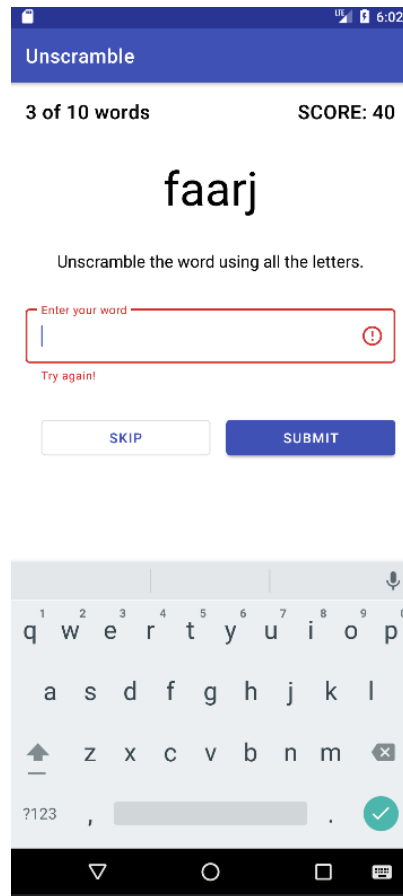
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	xmlns:tools="http://schemas.android.com/tools"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	tools:context=".MainActivity">
8	
9	<androidx.fragment.app.FragmentContainerView
10	android:id="@+id/game_fragment"
11	android:name="com.example.android.unscramble.ui.game.GameFragment"
12	android:layout_width="0dp"
13	android:layout_height="0dp"
14	app:layout_constraintBottom_toBottomOf="parent"
15	app:layout_constraintLeft_toLeftOf="parent"
16	app:layout_constraintRight_toRightOf="parent"
17	app:layout_constraintTop_toTopOf="parent" />
18	</androidx.constraintlayout.widget.ConstraintLayout>

Tabel 10. Source Code Soal 1 XML yang kedua

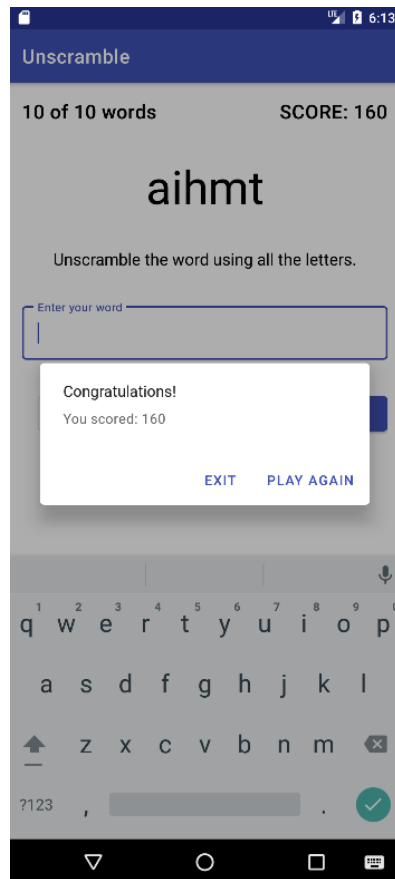
B. Output Program



Gambar 14. Screenshot Hasil Jawaban Soal 1 yang ke 1



Gambar 15. Screenshot Hasil Jawaban Soal 1 yang ke 2



Gambar 16. Screenshot Hasil Jawaban Soal 1 yang ke 3

C. Pembahasan

Untuk file GameFragment.kt

Pada baris ke 1, `package com.example.android.unscramble.ui.game:` Mendeklarasikan paket untuk kelas ini. Semua kelas dalam file ini berada dalam paket `com.example.android.unscramble.ui.game`.

Pada baris ke 3, `import android.os.Bundle:` Mengimpor kelas `Bundle` dari Android SDK, yang digunakan untuk menyimpan status instance.

Pada baris ke 4, `import android.view.LayoutInflater:` Mengimpor kelas `LayoutInflater` yang digunakan untuk mengembangkan (inflate) layout XML menjadi tampilan

Pada baris ke 5, `import android.view.View:` Mengimpor kelas `View` dari Android SDK, yang merupakan kelas dasar untuk semua elemen UI.

Pada baris ke 6, `import android.view.ViewGroup`: Mengimpor kelas `ViewGroup` yang merupakan kelas dasar untuk tata letak UI.

Pada baris ke 7, `import androidx.databinding.DataBindingUtil`: Mengimpor kelas `DataBindingUtil` yang digunakan untuk menghubungkan layout XML dengan data model.

Pada baris ke 8, `import androidx.fragment.app.Fragment`: Mengimpor kelas `Fragment` dari AndroidX library, yang menyediakan fitur untuk fragment.

Pada baris ke 9, `import androidx.fragment.app.viewModels`: Mengimpor ekstensi `viewModels` yang digunakan untuk mendapatkan instance `ViewModel`.

Pada baris ke 10, `import com.example.android.unscramble.R`: Mengimpor referensi sumber daya dari proyek ini.

Pada baris ke 11, `import com.example.android.unscramble.databinding.GameFragmentBinding`: Mengimpor kelas `GameFragmentBinding` yang dihasilkan secara otomatis dari file layout XML `game_fragment.xml`. Binding ini memungkinkan akses langsung ke tampilan dalam layout tanpa perlu menggunakan `findViewById`.

Pada baris ke 12, `import com.google.android.material.dialog.MaterialAlertDialogBuilder`: Mengimpor kelas `MaterialAlertDialogBuilder` dari Material Design library, yang digunakan untuk membuat dialog.

Pada baris ke 14, `class GameFragment : Fragment() {}`: Mendefinisikan kelas `GameFragment`, yang merupakan turunan dari `Fragment`.

Pada baris ke 16, `private lateinit var binding: GameFragmentBinding`: Mendeklarasikan variabel `binding` dengan tipe `GameFragmentBinding` menggunakan `lateinit`, yang berarti variabel ini akan diinisialisasi nanti sebelum digunakan.

Pada baris ke 18, `private val viewModel: GameViewModel by viewModels()`: Mendeklarasikan dan menginisialisasi properti `viewModel` dari tipe `GameViewModel` menggunakan delegasi `by viewModels()`, yang menyediakan instance `ViewModel` yang terkait dengan fragment ini.

Pada baris ke 20-23, `override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View {}`: Mendefinisikan metode `onCreateView` yang dipanggil untuk mengembang (inflate) dan mengembalikan hierarki tampilan yang terkait dengan fragment.

Pada baris ke 25, `binding = DataBindingUtil.inflate(inflater, R.layout.game_fragment, container, false)`: Menginisialisasi binding dengan mengembang (inflate) layout `game_fragment.xml` menggunakan `DataBindingUtil` dan menghubungkannya dengan fragment ini.

Pada baris ke 26, `return binding.root`: Mengembalikan root dari objek binding sebagai tampilan untuk fragment ini.

Pada baris ke 29, override fun `onViewCreated(view: View, savedInstanceState: Bundle?)` {}: Mendefinisikan metode `onViewCreated` yang dipanggil setelah hierarki tampilan telah diinflasi dan fragment siap digunakan.

Pada baris ke 30, `super.onViewCreated(view, savedInstanceState)`: Memanggil metode `onViewCreated` dari superclass untuk menjalankan inisialisasi dasar dari fragment.

Pada baris ke 32, `binding.gameViewModel = viewModel`: Mengikat `viewModel` ke variabel `gameViewModel` dalam layout, memungkinkan binding data antara `ViewModel` dan tampilan.

Pada baris ke 33, `binding.maxNoOfWords = MAX_NO_OF_WORDS`: Mengikat konstanta `MAX_NO_OF_WORDS` ke variabel `maxNoOfWords` dalam layout.

Pada baris ke 35, `binding.lifecycleOwner = viewLifecycleOwner`: Mengatur pemilik siklus hidup dari binding ke pemilik siklus hidup dari fragment ini, memungkinkan binding untuk mengamati `LiveData`.

Pada baris ke 37, `binding.submit.setOnClickListener { onSubmitWord() }`: Menetapkan listener `onClick` pada tombol `submit`. Saat tombol diklik, fungsi `onSubmitWord` akan dipanggil.

Pada baris ke 38, `binding.skip.setOnClickListener { onSkipWord() }`: Menetapkan listener `onClick` pada tombol `skip`. Saat tombol diklik, fungsi `onSkipWord` akan dipanggil.

Pada baris ke 41, private fun `onSubmitWord()` {}: Mendefinisikan metode `onSubmitWord` yang akan dipanggil saat tombol `submit` diklik.

Pada baris ke 42, `val playerWord = binding.textInputEditText.text.toString()`: Mengambil teks dari input `textInputEditText` dan menyimpannya sebagai `playerWord`.

Pada baris ke 44, `if (viewModel.isUserWordCorrect(playerWord))` {}: Memeriksa apakah kata yang dimasukkan pengguna benar menggunakan metode `isUserWordCorrect` dari `ViewModel`.

Pada baris ke 45, `setErrorTextField(false)`: Menonaktifkan tampilan error pada input teks jika kata benar.

Pada baris ke 46, `if (!viewModel.nextWord())` {: Memeriksa apakah ada kata berikutnya yang tersedia di `ViewModel`. Jika tidak, maka permainan selesai.

Pada baris ke 47, `showFinalScoreDialog()`: Menampilkan dialog skor akhir jika tidak ada kata berikutnya.

Pada baris ke 49, `} else` {: Blok else untuk kondisi jika kata yang dimasukkan pengguna salah.

Pada baris ke 50, `setErrorTextField(true)`: Mengaktifkan tampilan error pada input teks jika kata salah.

Pada baris ke 54, `private fun onSkipWord()` {: Mendefinisikan metode `onSkipWord` yang akan dipanggil saat tombol skip diklik.

Pada baris ke 55, `if (viewModel.nextWord())` {: Memeriksa apakah ada kata berikutnya yang tersedia di `ViewModel`.

Pada baris ke 56, `setErrorTextField(false)`: Menonaktifkan tampilan error pada input teks.

Pada baris ke 57, `} else` {: Blok else untuk kondisi jika tidak ada kata berikutnya.

Pada baris ke 58, `showFinalScoreDialog()`: Menampilkan dialog skor akhir jika tidak ada kata berikutnya.

Pada baris ke 62, `private fun showFinalScoreDialog()` {: Mendefinisikan metode `showFinalScoreDialog` yang akan menampilkan dialog skor akhir.

Pada baris ke 63, `MaterialAlertDialogBuilder(requireContext())`: Membuat instance `MaterialAlertDialogBuilder` dengan konteks dari fragment ini.

Pada baris ke 64, `.setTitle(getString(R.string.congratulations))`: Mengatur judul dialog dengan string `congratulations`.

Pada baris ke 65, `.setMessage(getString(R.string.you_scored, viewModel.score.value))`: Mengatur pesan dialog dengan skor yang diperoleh pengguna.

Pada baris ke 66, `.setCancelable(false)`: Mengatur dialog agar tidak bisa dibatalkan dengan menekan di luar dialog.

Pada baris ke 67, `.setNegativeButton(getString(R.string.exit)) { _, _ ->:` Menambahkan tombol negatif "Exit" yang memanggil metode `exitGame` saat diklik.

Pada baris ke 68, `exitGame():` Menutup aktivitas saat ini, mengakhiri permainan.

Pada baris ke 70, `.setPositiveButton(getString(R.string.play_again)) { _, _ ->:` Menambahkan tombol positif "Play Again" yang memanggil metode `restartGame` saat diklik.

Pada baris ke 71, `restartGame():` Memulai kembali permainan dengan mengatur ulang data di `ViewModel`.

Pada baris ke 73, `.show():` Menampilkan dialog kepada pengguna.

Pada baris ke 76, `private fun restartGame() {:` Mendefinisikan metode `restartGame` yang akan memulai ulang permainan.

Pada baris ke 77, `viewModel.reinitializeData():` Memanggil metode `reinitializeData` dari `ViewModel` untuk mengatur ulang data permainan.

Pada baris ke 78, `setErrorTextField(false):` Menonaktifkan tampilan error pada input teks.

Pada baris ke 81, `private fun exitGame() {:` Mendefinisikan metode `exitGame` yang akan mengakhiri permainan.

Pada baris ke 82 `activity?.finish():` Menutup aktivitas saat ini, mengakhiri permainan.

Pada baris ke 85, `private fun setErrorTextField(error: Boolean) {:` Mendefinisikan metode `setErrorTextField` yang akan mengatur tampilan error pada input teks.

Pada baris ke 86, `if (error) {:` Memeriksa apakah parameter error bernilai true.

Pada baris ke 87, `binding.textField.isEnabled = true:` Mengaktifkan tampilan error pada input teks.

Pada baris ke 88, `binding.textField.error = getString(R.string.try_again):` Menampilkan pesan error "Try Again".

Pada baris ke 89, `} else {:` Blok else untuk kondisi jika error bernilai false.

Pada baris ke 90, `binding.textField.isEnabled = false:` Menonaktifkan tampilan error pada input teks.

Pada baris ke 91, `binding.textInputEditText.text = null:` Mengosongkan teks dalam input teks.

Pada baris ke 92, `}`: Menutup metode `setErrorTextField`.

Pada baris ke 93, `}`: Menutup kelas `GameFragment`.

Untuk file `GameViewModel.kt`

Pada baris ke 1, `package com.example.android.unscramble.ui.game`: Mendeklarasikan paket untuk kelas ini. Semua kelas dalam file ini berada dalam paket `com.example.android.unscramble.ui.game`.

Pada baris ke 3, `import android.text.Spannable`: Mengimpor kelas `Spannable` dari Android SDK, yang memungkinkan pengaturan format teks yang dapat diedit.

Pada baris ke 4, `import android.text.SpannableString`: Mengimpor kelas `SpannableString` yang merupakan implementasi dari `Spannable`.

Pada baris ke 5, `import android.text.style.TtsSpan`: Mengimpor kelas `TtsSpan` yang digunakan untuk memberikan petunjuk teks ke-speech (TTS).

Pada baris ke 6, `import android.util.Log`: Mengimpor kelas `Log` untuk mencatat pesan debug.

Pada baris ke 7, `import androidx.lifecycle.LiveData`: Mengimpor kelas `LiveData` dari AndroidX library, yang digunakan untuk mengamati data.

Pada baris ke 8, `import androidx.lifecycle.MutableLiveData`: Mengimpor kelas `MutableLiveData` dari AndroidX library, yang memungkinkan data yang dapat berubah dan dapat diamati.

Pada baris ke 9, `import androidx.lifecycle.Transformations`: Mengimpor kelas `Transformations` yang menyediakan fungsi transformasi untuk `LiveData`.

Pada baris ke 10, `import androidx.lifecycle.ViewModel`: Mengimpor kelas `ViewModel` dari AndroidX library, yang merupakan kelas dasar untuk `ViewModel` di Android.

Pada baris ke 12, `class GameViewModel : ViewModel() {`: Mendefinisikan kelas `GameViewModel` yang merupakan turunan dari `ViewModel`.

Pada baris ke 13, `private val _score = MutableLiveData(0)`: Mendeklarasikan variabel `_score` dari tipe `MutableLiveData` dengan nilai awal 0. Variabel ini digunakan untuk menyimpan skor permainan.

Pada baris ke 14-15, `val score: LiveData<Int> get() = _score`: Mendeklarasikan properti `score` dari tipe `LiveData<Int>` yang hanya memiliki getter, mengembalikan nilai `_score`.

Pada baris ke 17, `private val _currentWordCount = MutableLiveData(0):` Mendeklarasikan variabel `_currentWordCount` dari tipe `MutableLiveData` dengan nilai awal 0. Variabel ini digunakan untuk menyimpan jumlah kata yang telah ditebak.

Pada baris ke 18, `val currentWordCount: LiveData<Int> get() = _currentWordCount:` Mendeklarasikan properti `currentWordCount` dari tipe `LiveData<Int>` yang hanya memiliki getter, mengembalikan nilai `_currentWordCount`.

Pada baris ke 21, `private val _currentScrambledWord = MutableLiveData<String>():` Mendeklarasikan variabel `_currentScrambledWord` dari tipe `MutableLiveData<String>`. Variabel ini digunakan untuk menyimpan kata yang diacak saat ini.

Pada baris ke 22, `val currentScrambledWord: LiveData<Spannable> = Transformations.map(_currentScrambledWord) {:` Mendeklarasikan properti `currentScrambledWord` dari tipe `LiveData<Spannable>` yang ditransformasikan dari `_currentScrambledWord`.

Pada baris ke 23, `if (it == null) {:` Memeriksa apakah nilai `_currentScrambledWord` adalah null.

Pada baris ke 24, `SpannableString(""):` Mengembalikan `SpannableString` kosong jika nilai `_currentScrambledWord` adalah null.

Pada baris ke 25, `else {:` Blok else untuk kondisi jika nilai `_currentScrambledWord` tidak null.

Pada baris ke 26, `val scrambledWord = it.toString():` Mengonversi nilai `_currentScrambledWord` menjadi string dan menyimpannya sebagai `scrambledWord`.

Pada baris ke 27, `val spannable: Spannable = SpannableString(scrambledWord):` Membuat instance `SpannableString` dari `scrambledWord`.

Pada baris ke 28-32, `spannable.setSpan(TtsSpan.VerbatimBuilder(scrambledWord).build(), 0, scrambledWord.length, Spannable.SPAN_INCLUSIVE_INCLUSIVE):` Menetapkan span TTS pada `spannable` untuk memberikan petunjuk TTS.

Pada baris ke 34, `spannable:` Mengembalikan `spannable` yang telah diatur.

Pada baris ke 35, `}`: Menutup blok else.

Pada baris ke 36, `}`: Menutup lambda transformasi.

Pada baris ke 38, `private var wordsList: MutableList<String> = mutableListOf()`: Mendeklarasikan variabel `wordsList` dari tipe `MutableList<String>` yang digunakan untuk menyimpan daftar kata yang telah digunakan dalam permainan.

Pada baris ke 39, `private lateinit var currentWord: String`: Mendeklarasikan variabel `currentWord` dengan tipe `String` menggunakan `lateinit`, yang berarti variabel ini akan diinisialisasi nanti sebelum digunakan.

Pada baris ke 41-43, `init { getNextWord() }`: Inisialisasi blok yang memanggil metode `getNextWord` saat `GameViewModel` dibuat.

Pada baris ke 45, `private fun getNextWord() {`: Mendefinisikan metode `getNextWord` yang digunakan untuk mengambil kata berikutnya.

Pada baris ke 46, `currentWord = allWordsList.random()`: Mengambil kata acak dari `allWordsList` dan menyimpannya sebagai `currentWord`.

Pada baris ke 47, `val tempWord = currentWord.toCharArray()`: Mengonversi `currentWord` menjadi array karakter dan menyimpannya sebagai `tempWord`.

Pada baris ke 48, `tempWord.shuffle()`: Mengacak urutan karakter dalam `tempWord`.

Pada baris ke 50, `while (String(tempWord).equals(currentWord, false)) { tempWord.shuffle() }`: Mengulangi pengacakan karakter dalam `tempWord` sampai karakter dalam `tempWord` berbeda dengan `currentWord`.

Pada baris ke 53, `if (wordsList.contains(currentWord)) { getNextWord() }`: Jika `currentWord` sudah ada dalam `wordsList`, memanggil `getNextWord` lagi untuk mendapatkan kata baru.

Pada baris ke 55, `else {`: Blok `else` untuk kondisi jika `currentWord` belum ada dalam `wordsList`.

Pada baris ke 56, `Log.d("Unscramble", "currentWord= $currentWord")`: Mencatat pesan debug yang menampilkan `currentWord`.

Pada baris ke 57, `_currentScrambledWord.value = String(tempWord)`: Mengatur nilai `_currentScrambledWord` dengan `tempWord` yang telah diacak.

Pada baris ke 58, `_currentWordCount.value = _currentWordCount.value?.inc()`: Meningkatkan nilai `_currentWordCount` dengan 1.

Pada baris ke 59, `wordsList.add(currentWord)`: Menambahkan `currentWord` ke dalam `wordsList`.

Pada baris ke 60, `}`: Menutup blok `else`.

Pada baris ke 61, `}`: Menutup metode `getNextWord`.

Pada baris ke 63, `fun reinitializeData() {`: Mendefinisikan metode `reinitializeData` yang digunakan untuk mengatur ulang data permainan.

Pada baris ke 64, `_score.value = 0`: Mengatur nilai `_score` menjadi 0.

Pada baris ke 65, `_currentWordCount.value = 0`: Mengatur nilai `_currentWordCount` menjadi 0.

Pada baris ke 66, `wordsList.clear()`: Mengosongkan `wordsList`.

Pada baris ke 67, `getNextWord()`: Memanggil metode `getNextWord` untuk memulai permainan baru dengan kata baru.

Pada baris ke 68, `}`: Menutup metode `reinitializeData`.

Pada baris ke 70, `private fun increaseScore() {`: Mendefinisikan metode `increaseScore` yang digunakan untuk meningkatkan skor.

Pada baris ke 71, `_score.value = _score.value?.plus(SCORE_INCREASE)`: Menambah nilai `_score` dengan `SCORE_INCREASE`.

Pada baris ke 72, `}`: Menutup metode `increaseScore`.

Pada baris ke 74, `fun isUserWordCorrect(playerWord: String): Boolean {`: Mendefinisikan metode `isUserWordCorrect` yang digunakan untuk memeriksa apakah kata yang dimasukkan pengguna benar.

Pada baris ke 75, `if (playerWord.equals(currentWord, true)) {`: Memeriksa apakah `playerWord` sama dengan `currentWord` (dengan mengabaikan huruf besar/kecil).

Pada baris ke 76, `increaseScore()`: Memanggil metode `increaseScore` untuk meningkatkan skor jika kata benar.

Pada baris ke 77, `return true`: Mengembalikan nilai `true` jika kata benar.

Pada baris ke 78, `}`: Menutup blok `if`.

Pada baris ke 79, `return false`: Mengembalikan nilai `false` jika kata salah.

Pada baris ke 80, `}`: Menutup metode `isUserWordCorrect`.

Pada baris ke 82, `fun nextWord(): Boolean {`: Mendefinisikan metode `nextWord` yang digunakan untuk mendapatkan kata berikutnya.

Pada baris ke 83, `return if (_currentWordCount.value!! < MAX_NO_OF_WORDS) {`: Memeriksa apakah nilai `_currentWordCount` kurang dari `MAX_NO_OF_WORDS`.

Pada baris ke 84, getNextWord(): Memanggil metode getNextWord untuk mendapatkan kata baru.

Pada baris ke 85, true: Mengembalikan nilai true jika masih ada kata berikutnya.

Pada baris ke 86, } else false: Mengembalikan nilai false jika tidak ada kata berikutnya.

Pada baris ke 87, }: Menutup metode nextWord.

Pada baris ke 88, }: Menutup kelas GameViewModel.

Untuk file ListofWords.kt

Pada baris ke 1, package com.example.android.unscramble.ui.game: Mendeklarasikan paket untuk file ini. Semua kelas dan objek dalam file ini berada dalam paket com.example.android.unscramble.ui.game.

Pada baris ke 4, const val MAX_NO_OF_WORDS = 10: Mendeklarasikan konstanta MAX_NO_OF_WORDS dengan nilai 10. Konstanta ini digunakan untuk menentukan jumlah maksimal kata dalam permainan.

Pada baris ke 5, const val SCORE_INCREASE = 20: Mendeklarasikan konstanta SCORE_INCREASE dengan nilai 20. Konstanta ini digunakan untuk menentukan jumlah peningkatan skor setiap kali pemain menebak kata dengan benar.

Pada baris ke 8, val allWordsList: List<String> =: Mendeklarasikan variabel allWordsList dari tipe List<String>. Variabel ini berisi daftar semua kata yang akan digunakan dalam permainan. Daftar ini tidak bisa diubah (immutable) karena menggunakan val.

Pada baris ke 9, listOf("apel", "angin", "anak", "asap", "asri", "awan", "ayam", "ajar", "amal", "adab"); Memulai daftar kata dengan beberapa contoh kata. listOf adalah fungsi yang membuat daftar yang tidak bisa diubah yang berisi elemen yang diberikan sebagai argumen.

Pada baris ke 10-36, adalah lanjutan daftar kata sama dengan baris ke 9.

Untuk File MainActivity.kt

Pada baris ke 1, package com.example.android.unscramble: Mendeklarasikan paket untuk file ini. Semua kelas dan objek dalam file ini berada dalam paket com.example.android.unscramble.

Pada baris ke 3, import android.os.Bundle: Mengimpor kelas Bundle dari paket android.os. Bundle digunakan untuk menyimpan dan memulihkan status aplikasi.

Pada baris ke 4, `import androidx.appcompat.app.AppCompatActivity`: Mengimpor kelas `AppCompatActivity` dari paket `androidx.appcompat.app`. `AppCompatActivity` adalah kelas dasar untuk aktivitas yang menggunakan fitur `ActionBar`.

Pada baris ke 9, `class MainActivity : AppCompatActivity() {}`: Mendeklarasikan kelas `MainActivity` yang merupakan subclass dari `AppCompatActivity`. Ini berarti `MainActivity` adalah sebuah aktivitas yang dapat menggunakan fitur-fitur dari `AppCompatActivity`.

Pada baris ke 11, `override fun onCreate(savedInstanceState: Bundle?) {}`: Mendefinisikan metode `onCreate` yang dipanggil ketika aktivitas pertama kali dibuat. Metode ini menggunakan parameter `savedInstanceState` untuk memulihkan status aktivitas jika sebelumnya telah disimpan.

Pada baris ke 12, `super.onCreate(savedInstanceState)`: Memanggil metode `onCreate` dari kelas induk (`AppCompatActivity`) untuk melakukan inisialisasi dasar aktivitas.

Pada baris ke 13, `setContentView(R.layout.main_activity)`: Menetapkan layout yang akan digunakan untuk aktivitas ini. Layout `R.layout.main_activity` diatur sebagai konten tampilan untuk aktivitas ini. `R.layout.main_activity` mengacu pada file XML layout yang terletak di folder `res/layout` dengan nama `main_activity.xml`.

Pada baris ke 14, `}`: Mengakhiri metode `onCreate`.

Pada baris ke 15, `}`: Mengakhiri kelas `MainActivity`.

Untuk file `game_fragment.xml`

Pada baris ke 1, `<?xml version="1.0" encoding="utf-8"?>`: Deklarasi XML ini menunjukkan bahwa dokumen ini menggunakan versi 1.0 dari standar XML dan encoding UTF-8. Ini adalah baris standar yang muncul di bagian atas hampir semua file XML.

Pada baris ke 3, `<layout xmlns:android="http://schemas.android.com/apk/res/android"` dan pada baris ke 4, `xmlns:app="http://schemas.android.com/apk/res-auto"` dan pada baris ke 5, `xmlns:tools="http://schemas.android.com/tools">`: Ini adalah tag root dari file XML ini, yang mendefinisikan elemen layout. Tag ini juga mendeklarasikan tiga namespace XML yang umum digunakan dalam pengembangan aplikasi Android: `xmlns:android="http://schemas.android.com/apk/res/android"`: Namespace utama untuk atribut standar Android. `xmlns:app="http://schemas.android.com/apk/res-auto"`: Namespace untuk atribut khusus yang dibuat oleh pengembang atau library pihak ketiga. `xmlns:tools="http://schemas.android.com/tools"`: Namespace untuk atribut alat bantu pengembangan yang hanya digunakan oleh Android Studio (seperti konteks desain).

Pada baris ke 7, `<data>`: Tag ini menandai awal dari blok data binding. Data binding adalah fitur yang memungkinkan Anda untuk mengikat UI komponen langsung ke data sumber dalam aplikasi Anda. Semua variabel dan impor yang akan digunakan dalam data binding didefinisikan di dalam tag `<data>` ini.

Pada baris ke 8-10, `<variable name="gameViewModel" type="com.example.android.unscramble.ui.game.GameViewModel" />`: Ini mendefinisikan sebuah variabel dengan nama `gameViewModel` yang merupakan objek dari kelas `GameViewModel`. Kelas ini terletak di paket `com.example.android.unscramble.ui.game`. Variabel ini dapat digunakan dalam layout untuk mengakses data dan fungsi dari `GameViewModel`.

Pada baris ke 11-13, `<variable name="maxNoOfWords" type="int" />`: Ini mendefinisikan sebuah variabel dengan nama `maxNoOfWords` yang merupakan tipe data integer. Variabel ini dapat digunakan dalam layout untuk mengakses jumlah maksimum kata.

Pada baris ke 14, `</data>`: Tag penutup untuk blok data binding. Semua variabel yang didefinisikan untuk data binding harus berada di antara tag pembuka `<data>` dan tag penutup `</data>`.

Pada baris ke 16, `<ScrollView>`: Ini adalah tag untuk menandai bahwa semua konten di dalamnya dapat di-scroll. `ScrollView` memungkinkan pengguna untuk menggulir konten jika ukurannya melebihi ukuran layar.

Pada baris ke 17, `android:layout_width="match_parent"`: Atribut ini mengatur lebar dari `ScrollView` agar sesuai dengan lebar layar (`match_parent`), sehingga akan mengambil sebanyak ruang yang tersedia di dalam parent layout.

Pada baris ke 18, `android:layout_height="match_parent"`: Atribut ini mengatur tinggi dari `ScrollView` agar sesuai dengan tinggi layar (`match_parent`), sehingga akan mengambil sebanyak ruang yang tersedia di dalam parent layout.

Pada baris ke 20, `<androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah tag untuk layout root di dalam `ScrollView`. `ConstraintLayout` adalah layout yang memungkinkan penggunaan constraint (batasan) antara elemen-elemen di dalamnya, memungkinkan penempatan yang fleksibel dan responsif.

Pada baris ke 21, `android:layout_width="match_parent"`: Atribut ini mengatur lebar dari `ConstraintLayout` agar sesuai dengan lebar layar (`match_parent`), sehingga akan mengambil sebanyak ruang yang tersedia di dalam parent layout.

Pada baris ke 22, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi dari `ConstraintLayout` agar sesuai dengan konten yang ada di dalamnya (`wrap_content`), sehingga akan mengubah tingginya sesuai dengan kebutuhan konten.

Pada baris ke 23, `android:padding="@dimen/default_padding"`: Atribut ini menambahkan padding di sekitar batas-batas `ConstraintLayout`. Nilai padding diambil dari `dimen default_padding`, yang merupakan nilai yang didefinisikan di file `dimens.xml`.

Pada baris ke 24, `tools:context=".ui.game.GameFragment"`: Atribut ini memberikan konteks untuk layout tersebut, yang berguna saat merancang tata letak dalam editor Android Studio. Di sini, konteksnya diatur ke `GameFragment`, sehingga saat merancang tampilan di Android Studio, editor dapat menampilkan preview yang sesuai dengan fragment ini.

Pada baris ke 26, `<Button>`: Ini adalah tag untuk menandai sebuah tombol dalam tata letak XML.

Pada baris ke 27, `android:id="@+id/skip"`: Ini adalah atribut yang memberikan ID unik untuk tombol. ID ini dapat digunakan untuk mengidentifikasi tombol secara unik dalam kode Java atau Kotlin.

Pada baris ke 28, `style="?attr/materialButtonOutlinedStyle"`: Atribut ini menentukan gaya atau tema yang akan diterapkan pada tombol. Nilai `?attr/materialButtonOutlinedStyle` menunjukkan bahwa tombol akan menggunakan gaya yang didefinisikan dalam tema aplikasi, yang mungkin disesuaikan dengan gaya `Material Design`.

Pada baris ke 29, `android:layout_width="0dp"`: Atribut ini mengatur lebar tombol menjadi `0dp`, yang berarti lebarnya akan diatur oleh constraint (batasan) yang didefinisikan di dalam `ConstraintLayout`.

Pada baris ke 30, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi tombol agar sesuai dengan tinggi konten di dalamnya.

Pada baris ke 31, `android:layout_marginStart="@dimen/default_padding"`: Atribut ini menentukan margin (jarak) di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) tombol. Nilai margin diambil dari `dimen default_padding`, yang merupakan nilai yang didefinisikan di file `dimens.xml`.

Pada baris ke 32, `android:layout_marginEnd="@dimen/default_padding"`: Atribut ini menentukan margin di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) tombol. Nilai margin juga diambil dari `dimen default_padding`.

Pada baris ke 33, `android:text="@string/skip"`: Atribut ini menentukan teks yang akan ditampilkan pada tombol. Nilai teks diambil dari string resource dengan nama "skip".

Pada baris ke 34, `app:layout_constraintBaseline_toBaselineOf="@+id/submit"`: Atribut ini menetapkan baseline (garis dasar) dari tombol ke baseline dari tata letak elemen lain yang ditentukan. Di sini, baseline tombol disesuaikan dengan baseline dari elemen dengan ID "submit".

Pada baris ke 35, `app:layout_constraintEnd_toStartOf="@+id/submit"`: Atribut ini menetapkan constraint (batasan) di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) tombol ke sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) elemen dengan ID "submit". Ini mengatur jarak horizontal antara tombol "skip" dan tombol "submit".

Pada baris ke 36, `app:layout_constraintStart_toStartOf="parent"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) tombol ke sisi awal dari parent layout. Ini memastikan tombol "skip" terletak di sebelah kiri (ltr) atau kanan (rtl) dari parent layout.

Pada baris ke 38, `<Button>`: Ini adalah tag untuk menandai sebuah tombol dalam tata letak XML.

Pada baris ke 39, `android:id="@+id/submit"`: Ini adalah atribut yang memberikan ID unik untuk tombol. ID ini dapat digunakan untuk mengidentifikasi tombol secara unik dalam kode Java atau Kotlin.

Pada baris ke 40, `android:layout_width="0dp"`: Atribut ini mengatur lebar tombol menjadi 0dp, yang berarti lebarnya akan diatur oleh constraint (batasan) yang didefinisikan di dalam ConstraintLayout.

Pada baris ke 41, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi tombol agar sesuai dengan tinggi konten di dalamnya.

Pada baris ke 42, `android:layout_marginTop="@dimen/default_margin"`: Atribut ini menentukan margin (jarak) di bagian atas tombol. Nilai margin diambil dari `dimen/default_margin`, yang merupakan nilai yang didefinisikan di file `dimens.xml`.

Pada baris ke 43, `android:text="@string/submit"`: Atribut ini menentukan teks yang akan ditampilkan pada tombol. Nilai teks diambil dari string resource dengan nama "submit".

Pada baris ke 44, `app:layout_constraintEnd_toEndOf="parent"`: Atribut ini menetapkan constraint (batasan) di sisi akhir (kanan dalam bahasa ltr atau kiri dalam

bahasa rtl) tombol ke sisi akhir dari parent layout. Ini memastikan tombol "submit" terletak di sebelah kanan (ltr) atau kiri (rtl) dari parent layout.

Pada baris ke 45, `app:layout_constraintStart_toEndOf="@+id/skip"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) tombol ke sisi akhir dari tombol dengan ID "skip". Ini memastikan tombol "submit" terletak di sebelah kanan (ltr) atau kiri (rtl) dari tombol "skip".

Pada baris ke 46, `app:layout_constraintTop_toBottomOf="@+id/textField"`: Atribut ini menetapkan constraint di bagian atas tombol ke bagian bawah dari elemen dengan ID "textField". Ini memastikan bahwa tombol "submit" berada di bawah (vertikal) dari elemen "textField".

Pada baris ke 48, `<TextView>`: Ini adalah tag untuk menandai sebuah TextView dalam tata letak XML.

Pada baris ke 49, `android:id="@+id/textView_instructions"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini dapat digunakan untuk mengidentifikasi TextView secara unik dalam kode Java atau Kotlin.

Pada baris ke 50, `android:layout_width="wrap_content"`: Atribut ini mengatur lebar TextView agar menyesuaikan dengan lebar konten di dalamnya.

Pada baris ke 51, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi TextView agar menyesuaikan dengan tinggi konten di dalamnya.

Pada baris ke 52, `android:text="@string/instructions"`: Atribut ini menentukan teks yang akan ditampilkan dalam TextView. Nilai teks diambil dari string resource dengan nama "instructions".

Pada baris ke 53, `android:textSize="17sp"`: Atribut ini menentukan ukuran teks dalam sp (scalable pixels). Di sini, ukuran teks diatur menjadi 17sp.

Pada baris ke 54, `app:layout_constraintBottom_toTopOf="@+id/textField"`: Atribut ini menetapkan constraint (batasan) di bagian bawah TextView ke bagian atas dari elemen dengan ID "textField". Ini memastikan bahwa TextView berada di atas (vertikal) dari elemen "textField".

Pada baris ke 55, `app:layout_constraintEnd_toEndOf="parent"`: Atribut ini menetapkan constraint di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) TextView ke sisi akhir dari parent layout. Ini memastikan bahwa TextView berada di sebelah kanan (ltr) atau kiri (rtl) dari parent layout.

Pada baris ke 56, `app:layout_constraintStart_toStartOf="parent"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl)

TextView ke sisi awal dari parent layout. Ini memastikan bahwa TextView berada di sebelah kiri (ltr) atau kanan (rtl) dari parent layout.

Pada baris ke 57,

`app:layout_constraintTop_toBottomOf="@+id/textView_unscrambled_word":`
Atribut ini menetapkan constraint di bagian atas TextView ke bagian bawah dari elemen dengan ID "textView_unscrambled_word". Ini memastikan bahwa TextView berada di bawah (vertikal) dari elemen "textView_unscrambled_word".

Pada baris ke 59, `<TextView>`: Ini adalah tag untuk menandai sebuah TextView dalam tata letak XML.

Pada baris ke 60, `android:id="@+id/textView_unscrambled_word"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini dapat digunakan untuk mengidentifikasi TextView secara unik dalam kode Java atau Kotlin.

Pada baris ke 61, `android:layout_width="wrap_content"`: Atribut ini mengatur lebar TextView agar menyesuaikan dengan lebar konten di dalamnya.

Pada baris ke 62, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi TextView agar menyesuaikan dengan tinggi konten di dalamnya.

Pada baris ke 63, `android:layout_marginTop="@dimen/default_margin"`: Atribut ini menentukan margin (jarak) di bagian atas TextView. Nilai margin diambil dari `dimen/default_margin`, yang merupakan nilai yang didefinisikan di file `dimens.xml`.

Pada baris ke 64, `android:layout_marginBottom="@dimen/default_margin"`: Atribut ini menentukan margin di bagian bawah TextView. Nilai margin juga diambil dari `dimen/default_margin`.

Pada baris ke 65, `android:text="@{gameViewModel.currentScrambledWord}"`:
Atribut ini menentukan teks yang akan ditampilkan dalam TextView. Teks ini diambil dari property `currentScrambledWord` yang terkait dengan objek `gameViewModel`. Ini adalah contoh penggunaan data binding di mana nilai teks TextView dihubungkan langsung dengan nilai dari properti `currentScrambledWord` pada objek `gameViewModel`.

Pada baris ke 66,

`android:textAppearance="@style/TextAppearance.MaterialComponents.Headline3"`:
Atribut ini menentukan penampilan teks TextView, yaitu menggunakan gaya teks yang ditentukan dalam gaya `TextAppearance.MaterialComponents.Headline3`.

Pada baris ke 67,

`app:layout_constraintBottom_toTopOf="@+id/textView_instructions"`: Atribut ini

menetapkan constraint (batasan) di bagian bawah TextView ke bagian atas dari elemen dengan ID "textView_instructions". Ini memastikan bahwa TextView berada di atas (vertikal) dari elemen "textView_instructions".

Pada baris ke 68, `app:layout_constraintEnd_toEndOf="parent"`: Atribut ini menetapkan constraint di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) TextView ke sisi akhir dari parent layout. Ini memastikan bahwa TextView berada di sebelah kanan (ltr) atau kiri (rtl) dari parent layout.

Pada baris ke 69, `app:layout_constraintStart_toStartOf="parent"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) TextView ke sisi awal dari parent layout. Ini memastikan bahwa TextView berada di sebelah kiri (ltr) atau kanan (rtl) dari parent layout.

Pada baris ke 70, `app:layout_constraintTop_toBottomOf="@+id/word_count"`: Atribut ini menetapkan constraint di bagian atas TextView ke bagian bawah dari elemen dengan ID "word_count". Ini memastikan bahwa TextView berada di bawah (vertikal) dari elemen "word_count".

Pada baris ke 71, `tools:text="Scramble word"`: Atribut ini hanya berlaku saat merancang tata letak dalam editor XML di Android Studio. Ini menampilkan teks dummy "Scramble word" dalam tampilan editor.

Pada baris ke 73, `<TextView>`: Ini adalah tag untuk menandai sebuah TextView dalam tata letak XML.

Pada baris ke 74, `android:id="@+id/word_count"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini dapat digunakan untuk mengidentifikasi TextView secara unik dalam kode Java atau Kotlin.

Pada baris ke 75, `android:layout_width="wrap_content"`: Atribut ini mengatur lebar TextView agar menyesuaikan dengan lebar konten di dalamnya.

Pada baris ke 76, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi TextView agar menyesuaikan dengan tinggi konten di dalamnya.

Pada baris ke 77, `android:text="@{ @string/word_count(gameViewModel.currentWordCount, maxNoOfWords) }"`: Atribut ini menentukan teks yang akan ditampilkan dalam TextView. Ini menggunakan data binding dengan ekspresi yang mengikat nilai teks ke sebuah string resource. Ekspresi tersebut akan mengevaluasi fungsi `word_count` yang didefinisikan dalam file string resource XML, yang menerima dua parameter: `gameViewModel.currentWordCount` dan `maxNoOfWords`. Ini memungkinkan

penyesuaian dinamis dari teks sesuai dengan nilai-nilai yang dihitung dalam kode aplikasi.

Pada baris ke 78,

`android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"`: Atribut ini menentukan penampilan teks TextView, yaitu menggunakan gaya teks yang ditentukan dalam gaya `TextAppearance.MaterialComponents.Headline6`.

Pada baris ke 79,

`app:layout_constraintBottom_toTopOf="@+id/textView_unscrambled_word"`: Atribut ini menetapkan constraint (batasan) di bagian bawah TextView ke bagian atas dari elemen dengan ID `"textView_unscrambled_word"`. Ini memastikan bahwa TextView berada di atas (vertikal) dari elemen `"textView_unscrambled_word"`.

Pada baris ke 80, `app:layout_constraintStart_toStartOf="parent"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) TextView ke sisi awal dari parent layout. Ini memastikan bahwa TextView berada di sebelah kiri (ltr) atau kanan (rtl) dari parent layout.

Pada baris ke 81, `app:layout_constraintTop_toTopOf="parent"`: Atribut ini menetapkan constraint di bagian atas TextView ke bagian atas dari parent layout. Ini memastikan bahwa TextView berada di bagian atas dari parent layout.

Pada baris ke 82, `tools:text="3 of 10 words"`: Atribut ini hanya berlaku saat merancang tata letak dalam editor XML di Android Studio. Ini menampilkan teks dummy `"3 of 10 words"` dalam tampilan editor.

Pada baris ke 84, `<TextView>`: Ini adalah tag untuk menandai sebuah TextView dalam tata letak XML.

Pada baris ke 85, `android:id="@+id/score"`: Ini adalah atribut yang memberikan ID unik untuk TextView. ID ini dapat digunakan untuk mengidentifikasi TextView secara unik dalam kode Java atau Kotlin.

Pada baris ke 86, `android:layout_width="wrap_content"`: Atribut ini mengatur lebar TextView agar menyesuaikan dengan lebar konten di dalamnya.

Pada baris ke 87, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi TextView agar menyesuaikan dengan tinggi konten di dalamnya.

Pada baris ke 88, `android:text="@{ @string/score(gameViewModel.score) }"`: Atribut ini menentukan teks yang akan ditampilkan dalam TextView. Ini menggunakan data binding dengan ekspresi yang mengikat nilai teks ke sebuah string resource. Ekspresi tersebut akan mengevaluasi fungsi `score` yang didefinisikan dalam file string resource

XML, yang menerima satu parameter: `gameViewModel.score`. Ini memungkinkan penyesuaian dinamis dari teks sesuai dengan nilai yang dihitung dalam kode aplikasi.

Pada baris ke 89, `android:textAllCaps="true"`: Atribut ini mengatur agar semua huruf dalam teks `TextView` menjadi huruf kapital.

Pada baris ke 90,

`android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"`: Atribut ini menentukan penampilan teks `TextView`, yaitu menggunakan gaya teks yang ditentukan dalam gaya `TextAppearance.MaterialComponents.Headline6`.

Pada baris ke 91, `app:layout_constraintEnd_toEndOf="parent"`: Atribut ini menetapkan constraint (batasan) di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) `TextView` ke sisi akhir dari parent layout. Ini memastikan bahwa `TextView` berada di sebelah kanan (ltr) atau kiri (rtl) dari parent layout.

Pada baris ke 92, `app:layout_constraintTop_toTopOf="parent"`: Atribut ini menetapkan constraint di bagian atas `TextView` ke bagian atas dari parent layout. Ini memastikan bahwa `TextView` berada di bagian atas dari parent layout.

Pada baris ke 93, `tools:text="Score: 20"`: Atribut ini hanya berlaku saat merancang tata letak dalam editor XML di Android Studio. Ini menampilkan teks dummy "Score: 20" dalam tampilan editor.

Pada baris ke 95, `<com.google.android.material.textfield.TextInputLayout>`: Ini adalah tag untuk menandai sebuah `TextInputLayout`, yang merupakan wadah untuk komponen input seperti `EditText` dalam tata letak XML.

Pada baris ke 96, `android:id="@+id/textField"`: Ini adalah atribut yang memberikan ID unik untuk `TextInputLayout`. ID ini dapat digunakan untuk mengidentifikasi `TextInputLayout` secara unik dalam kode Java atau Kotlin.

Pada baris ke 97, `style="@style/Widget.Unscramble.TextInputLayout.OutlinedBox"`: Atribut ini menentukan gaya atau tema untuk `TextInputLayout`. Di sini, gaya "OutlinedBox" yang ditentukan dalam file `styles.xml` dengan nama "Widget.Unscramble.TextInputLayout" diterapkan.

Pada baris ke 98, `android:layout_width="0dp"`: Atribut ini mengatur lebar `TextInputLayout` agar dihitung oleh constraint (batasan) yang didefinisikan di dalam `ConstraintLayout`.

Pada baris ke 99, `android:layout_height="wrap_content"`: Atribut ini mengatur tinggi `TextInputLayout` agar menyesuaikan dengan tinggi konten di dalamnya.

Pada baris ke 100, `android:layout_marginTop="@dimen/default_margin"`: Atribut ini menentukan margin (jarak) di bagian atas `TextInputLayout`. Nilai margin diambil dari `dimen default_margin`, yang merupakan nilai yang didefinisikan di file `dimens.xml`.

Pada baris ke 101, `android:hint="@string/enter_your_word"`: Atribut ini menentukan hint (petunjuk) yang akan ditampilkan dalam `TextInputLayout`. Nilai hint diambil dari string resource dengan nama `"enter_your_word"`.

Pada baris ke 102, `app:errorIconDrawable="@drawable/ic_error"`: Atribut ini menentukan gambar yang akan digunakan sebagai ikon kesalahan dalam `TextInputLayout`. Nilai gambar diambil dari drawable resource dengan nama `"ic_error"`.

Pada baris ke 103, `app:helperTextTextAppearance="@style/TextAppearance.MaterialComponents.Subtitle1"`: Atribut ini menentukan penampilan teks bantuan (helper text) dalam `TextInputLayout`. Di sini, gaya teks `"Subtitle1"` yang ditentukan dalam file `styles.xml` diterapkan.

Pada baris ke 104, `app:layout_constraintBottom_toTopOf="@+id/submit"`: Atribut ini menetapkan constraint (batasan) di bagian bawah `TextInputLayout` ke bagian atas dari elemen dengan ID `"submit"`. Ini memastikan bahwa `TextInputLayout` berada di atas (vertikal) dari elemen `"submit"`.

Pada baris ke 105, `app:layout_constraintEnd_toEndOf="parent"`: Atribut ini menetapkan constraint di sisi akhir (kanan dalam bahasa ltr atau kiri dalam bahasa rtl) `TextInputLayout` ke sisi akhir dari parent layout. Ini memastikan bahwa `TextInputLayout` berada di sebelah kanan (ltr) atau kiri (rtl) dari parent layout.

Pada baris ke 106, `app:layout_constraintStart_toStartOf="parent"`: Atribut ini menetapkan constraint di sisi awal (kiri dalam bahasa ltr atau kanan dalam bahasa rtl) `TextInputLayout` ke sisi awal dari parent layout. Ini memastikan bahwa `TextInputLayout` berada di sebelah kiri (ltr) atau kanan (rtl) dari parent layout.

Pada baris ke 107, `app:layout_constraintTop_toBottomOf="@+id/textView_instructions"`: Atribut ini menetapkan constraint di bagian atas `TextInputLayout` ke bagian bawah dari elemen dengan ID `"textView_instructions"`. Ini memastikan bahwa `TextInputLayout` berada di bawah (vertikal) dari elemen `"textView_instructions"`.

Pada baris ke 109, `<com.google.android.material.textfield.TextInputEditText>`: Ini adalah tag untuk menandai sebuah `EditText` yang terletak di dalam `TextInputLayout`. Ini digunakan untuk memasukkan dan menampilkan teks dalam `TextInputLayout`.

Pada baris ke 110, `android:id="@+id/text_input_edit_text"`: Ini adalah atribut yang memberikan ID unik untuk EditText. ID ini dapat digunakan untuk mengidentifikasi EditText secara unik dalam kode Java atau Kotlin.

Pada baris ke 111, `android:layout_width="match_parent"`: Atribut ini mengatur lebar EditText agar mengisi lebar penuh parent layout yang mengelilinginya.

Pada baris ke 112, `android:layout_height="match_parent"`: Atribut ini mengatur tinggi EditText agar mengisi tinggi penuh parent layout yang mengelilinginya.

Pada baris ke 113, `android:inputType="textPersonName|textNoSuggestions"`: Atribut ini menentukan jenis input yang diharapkan dari pengguna. Di sini, `inputType` disetel untuk menerima nama orang (`textPersonName`) dan tidak menampilkan saran teks (`textNoSuggestions`).

Pada baris ke 114, `android:maxLines="1"`: Atribut ini menentukan jumlah baris maksimum yang diperbolehkan di EditText. Di sini, disetel untuk satu baris, sehingga EditText hanya akan menampilkan satu baris teks.

Pada baris ke 115, `</com.google.android.material.textfield.TextInputLayout>`: Ini adalah penutup untuk tag `TextInputLayout` yang membatasi `TextInputEditText` di dalamnya. Ini menandai akhir dari `TextInputLayout` dan mengakhiri konfigurasi dan tata letak untuk komponen input.

Pada baris ke 117, `</androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah penutup untuk tag `ConstraintLayout` yang membatasi semua komponen tata letak di dalamnya. Ini menandai akhir dari `ConstraintLayout` dan mengakhiri konfigurasi dan tata letak untuk semua elemen di dalamnya.

Pada baris ke 118, `</ScrollView>`: Ini adalah penutup untuk tag `ScrollView` yang membatasi semua konten yang dapat di-scroll. Ini menandai akhir dari `ScrollView` dan mengakhiri konfigurasi dan tata letak untuk konten yang dapat di-scroll.

Pada baris ke 119, `</layout>`: Ini adalah penutup untuk tag `layout` utama yang mengelilingi semua elemen XML. Ini menandai akhir dari file layout XML dan mengakhiri konfigurasi dan tata letak untuk seluruh tata letak XML.

Untuk File `main_activity.xml`

Pada baris ke 1, `<?xml version="1.0" encoding="utf-8"?>`: Ini adalah deklarasi XML yang menunjukkan versi XML yang digunakan dan pengkodean karakter yang digunakan dalam dokumen.

Pada baris ke 2, `<androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah tag root yang menandakan bahwa layout ini menggunakan `ConstraintLayout` sebagai tata

letak induk. `xmlns:android="http://schemas.android.com/apk/res/android"`: Ini adalah deklarasi namespace untuk elemen XML yang berasal dari kerangka kerja Android.

Pada baris ke 3, `xmlns:app="http://schemas.android.com/apk/res-auto"`: Ini adalah deklarasi namespace untuk atribut khusus aplikasi yang didefinisikan dalam XML dan digunakan oleh alat Android Studio untuk menghasilkan kode Java yang sesuai.

Pada baris ke 4, `xmlns:tools="http://schemas.android.com/tools"`: Ini adalah deklarasi namespace untuk atribut alat yang hanya digunakan saat merancang tata letak dalam Android Studio, seperti preview tata letak dan properti yang disematkan.

Pada baris ke 5, `android:layout_width="match_parent"`: Atribut ini mengatur lebar layout agar mengisi lebar penuh dari parent-nya.

Pada baris ke 6, `android:layout_height="match_parent"`: Atribut ini mengatur tinggi layout agar mengisi tinggi penuh dari parent-nya.

Pada baris ke 7, `tools:context=".MainActivity"`: Atribut ini menentukan aktivitas yang terkait dengan layout ini saat dirancang dalam Android Studio. Ini membantu Android Studio dalam menampilkan preview tata letak yang akurat.

Pada baris ke 9, `<androidx.fragment.app.FragmentContainerView>`: Ini adalah tag yang menandakan bahwa ini adalah sebuah `FragmentContainerView`. `FragmentContainerView` adalah wadah untuk menampilkan fragmen dalam tata letak.

Pada baris ke 10, `android:id="@+id/game_fragment"`: Ini adalah atribut yang memberikan ID unik untuk `FragmentContainerView`. ID ini dapat digunakan untuk mengidentifikasi `FragmentContainerView` secara unik dalam kode Java atau Kotlin.

Pada baris ke 11, `android:name="com.example.android.unscramble.ui.game.GameFragment"`: Ini adalah atribut yang menentukan nama kelas fragment yang akan dimuat ke dalam `FragmentContainerView`. Di sini, fragment yang akan dimuat adalah `GameFragment` yang terletak di package `com.example.android.unscramble.ui.game`.

Pada baris ke 12, `android:layout_width="0dp"`: Atribut ini mengatur lebar `FragmentContainerView` agar dihitung oleh constraint (batasan) yang didefinisikan di dalam `ConstraintLayout`.

Pada baris ke 13, `android:layout_height="0dp"`: Atribut ini mengatur tinggi `FragmentContainerView` agar dihitung oleh constraint (batasan) yang didefinisikan di dalam `ConstraintLayout`.

Pada baris ke 14, `app:layout_constraintBottom_toBottomOf="parent"`: Atribut ini menetapkan constraint (batasan) di bagian bawah `FragmentContainerView` ke bagian

bawah dari parent layout. Ini memastikan bahwa FragmentContainerView berada di bagian bawah dari parent layout.

Pada baris ke 15, `app:layout_constraintLeft_toLeftOf="parent"`: Atribut ini menetapkan constraint di sisi kiri FragmentContainerView ke sisi kiri dari parent layout. Ini memastikan bahwa FragmentContainerView berada di sisi kiri dari parent layout.

Pada baris ke 16, `app:layout_constraintRight_toRightOf="parent"`: Atribut ini menetapkan constraint di sisi kanan FragmentContainerView ke sisi kanan dari parent layout. Ini memastikan bahwa FragmentContainerView berada di sisi kanan dari parent layout.

Pada baris ke 17, `app:layout_constraintTop_toTopOf="parent"`: Atribut ini menetapkan constraint di bagian atas FragmentContainerView ke bagian atas dari parent layout. Ini memastikan bahwa FragmentContainerView berada di bagian atas dari parent layout.

Pada baris ke 18, `</androidx.constraintlayout.widget.ConstraintLayout>`: Ini adalah tag penutup untuk ConstraintLayout yang mengakhiri definisi layout XML. Semua elemen yang ditata dalam ConstraintLayout sebelumnya harus berada di dalam tag pembuka dan penutup ini.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/farlyhaydyhdjalil/Praktikum-PEMROGRAMANMOBILE/tree/e3333d0e6f2df3dc0ea1cf52f0eb3ff0a52375cc/Modul%203/UnscrambleApp>

MODUL 4 : CONNECT TO THE INTERNET

SOAL 1

Soal Praktikum:

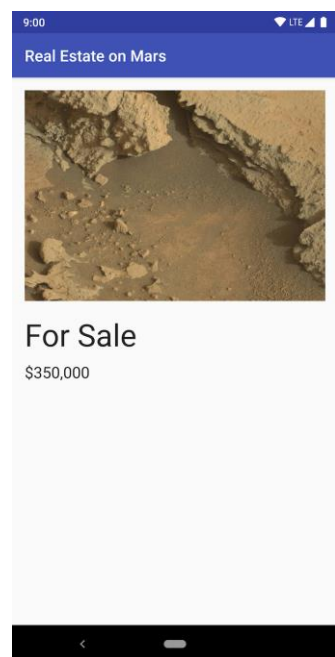
1. Buatlah sebuah aplikasi Android sederhana untuk menampilkan data dari Internet melalui Public API

1. Daftar Public API yang dapat digunakan dapat dilihat pada link berikut: <https://github.com/public-apis/public-apis> (dapat juga mengambil diluar dari link tersebut)
2. Pada saat dijalankan, aplikasi akan terhubung dengan Internet untuk menarik data dari **Public API** tersebut
3. Gunakan library tambahan yaitu **Retrofit** untuk mempermudah proses koneksi internet
4. Gunakan library tambahan yaitu **Moshi** untuk mempermudah proses data JSON
5. Gunakan library tambahan yaitu **Glide** untuk memuat dan menyimpan gambar dalam cache berdasarkan URL (opsional)
6. Data tersebut kemudian ditampilkan dalam bentuk **RecyclerView**
7. Masing-masing data di RecyclerView tersebut dapat diklik untuk menampilkan detailnya
8. Gunakan **LiveData** dan **ViewModel** untuk mempertahankan state dari aplikasi pada saat Configuration Changes
9. Saat pengguna merotasi tampilan handphone dari Portrait menjadi Landscape maka tampilan data yang sudah ada tidak boleh hilang

Contoh aplikasi:



Gambar 17. Contoh gambar pertama



Gambar 18. Contoh gambar kedua

A. Source Code

DetailFragment.kt

```
1  /*
2   * Copyright 2018, The Android Open Source Project
3   *
4   * Licensed under the Apache License, Version 2.0
5   * (the "License");
6   * you may not use this file except in compliance
7   * with the License.
8   * You may obtain a copy of the License at
9   *
10  *      http://www.apache.org/licenses/LICENSE-2.0
11  *
12  * Unless required by applicable law or agreed to
13  * in writing, software
14  * distributed under the License is distributed on
15  * an "AS IS" BASIS,
16  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
17  * either express or implied.
18  * See the License for the specific language
19  * governing permissions and
20  * limitations under the License.
21  */
22
23 package com.example.android.marsrealestate.detail
24
25 import android.os.Bundle
26 import android.view.LayoutInflater
27 import android.view.View
28 import android.view.ViewGroup
29 import androidx.fragment.app.Fragment
30 import androidx.lifecycle.ViewModelProvider
31 import
32     com.example.android.marsrealestate.databinding.FragmentDetailBinding
33
34 /**
35  * This [Fragment] shows the detailed information
36  * about a selected piece of Mars real estate.
37  * It sets this information in the
38  * [DetailViewModel], which it gets as a Parcelable
39  * property
```



```

30  * through Jetpack Navigation's SafeArgs.
31  */
32  class DetailFragment : Fragment() {
33      override fun onCreateView(inflater:
LayoutInflater, container: ViewGroup?,
34                              savedInstanceState:
Bundle?): View? {
35          val application =
requireNotNull(activity).application
36          val binding =
FragmentDetailBinding.inflate(inflater)
37          binding.lifecycleOwner = this
38          val marsProperty =
DetailFragmentArgs.fromBundle(arguments!!).selectedP
roperty
39          val viewModelFactory =
DetailViewModelFactory(marsProperty, application)
40          binding.viewModel = ViewModelProvider(
41              this,
viewModelFactory).get(DetailViewModel::class.java)
42          return binding.root
43      }
44  }

```

Tabel 11. Source Code Soal 1 Kotlin yang pertama

DetailViewModel.kt

```

1  /*
2  *   Copyright 2018, The Android Open Source Project
3  *
4  *   Licensed under the Apache License, Version 2.0
(the "License");
5  *   you may not use this file except in compliance
with the License.
6  *   You may obtain a copy of the License at
7  *
8  *       http://www.apache.org/licenses/LICENSE-2.0
9  *
10 *   Unless required by applicable law or agreed to
in writing, software
11 *   distributed under the License is distributed on
an "AS IS" BASIS,
12 *   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied.

```

```

13  * See the License for the specific language
    governing permissions and
14  * limitations under the License.
15  */
16
17 package com.example.android.marsrealestate.detail
18
19 import android.app.Application
20 import androidx.lifecycle.AndroidViewModel
21 import androidx.lifecycle.LiveData
22 import androidx.lifecycle.MutableLiveData
23 import androidx.lifecycle.Transformations
24 import androidx.lifecycle.ViewModel
25 import com.example.android.marsrealestate.R
26 import
    com.example.android.marsrealestate.network.MarsPrope
    rty
27
28 /**
29  * The [ViewModel] associated with the
    [DetailFragment], containing information about the
    selected
30  * [MarsProperty].
31  */
32 class DetailViewModel(marsProperty: MarsProperty,
    app: Application) : AndroidViewModel(app) {
33     private val _selectedProperty =
    MutableLiveData<MarsProperty>()
34
35     // The external LiveData for the
    SelectedProperty
36     val selectedProperty: LiveData<MarsProperty>
37     get() = _selectedProperty
38
39     // Initialize the _selectedProperty
    MutableLiveData
40     init {
41         _selectedProperty.value = marsProperty
42     }
43
44     // The displayPropertyPrice formatted
    Transformation Map LiveData, which displays the sale
45     // or rental price.

```

```

46     val displayPropertyPrice =
Transformations.map(selectedProperty) {
47         app.applicationContext.getString(
48             when (it.isRental) {
49                 true ->
R.string.display_price_monthly_rental
50                 false -> R.string.display_price
51             }, it.price)
52     }
53
54     // The displayPropertyType formatted
Transformation Map LiveData, which displays the
55     // "For Rent/Sale" String
56     val displayPropertyType =
Transformations.map(selectedProperty) {
57 app.applicationContext.getString(R.string.display_ty
pe,
58         app.applicationContext.getString(
59             when(it.isRental) {
60                 true -> R.string.type_rent
61                 false -> R.string.type_sale
62             })
63     }
64 }

```

Tabel 12. Source Code Soal 1 Kotlin yang kedua

DetailViewModelFactory.kt

```

1  /*
2   * Copyright 2018, The Android Open Source Project
3   *
4   * Licensed under the Apache License, Version 2.0
(the "License");
5   * you may not use this file except in compliance
with the License.
6   * You may obtain a copy of the License at
7   *
8   *     http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to
in writing, software
11  * distributed under the License is distributed on
an "AS IS" BASIS,

```

```

12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
    either express or implied.
13  * See the License for the specific language
    governing permissions and
14  * limitations under the License.
15  */
16
17 package com.example.android.marsrealestate.detail
18
19 import android.app.Application
20 import androidx.lifecycle.ViewModel
21 import androidx.lifecycle.ViewModelProvider
22 import
    com.example.android.marsrealestate.network.MarsPropert
    y
23
24 /**
25  * Simple ViewModel factory that provides the
    MarsProperty and context to the ViewModel.
26  */
27 class DetailViewModelFactory(
28     private val marsProperty: MarsProperty,
29     private val application: Application) :
    ViewModelProvider.Factory {
30     @Suppress("unchecked_cast")
31     override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
32         if
33         (modelClass.isAssignableFrom(DetailViewModel::class.
    java)) {
34             return DetailViewModel(marsProperty,
    application) as T
35         }
36         throw IllegalArgumentException("Unknown
    ViewModel class")
37     }

```

Tabel 13. Source Code Soal 1 Kotlin yang ketiga

MarsApiService.kt

```

1  /*
2   * Copyright 2018, The Android Open Source Project
3   *
4   * Licensed under the Apache License, Version 2.0
5   * (the "License");
6   * you may not use this file except in compliance
7   * with the License.
8   * You may obtain a copy of the License at
9   *
10  *      http://www.apache.org/licenses/LICENSE-2.0
11  *
12  * Unless required by applicable law or agreed to in
13  * writing, software
14  * distributed under the License is distributed on an
15  * "AS IS" BASIS,
16  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
17  * either express or implied.
18  * See the License for the specific language
19  * governing permissions and
20  * limitations under the License.
21  */
22
23 package com.example.android.marsrealestate.network
24
25 // import
26 com.jakewharton.retrofit2.adapter.kotlin.coroutines.C
27 oroutineCallAdapterFactory
28 import com.squareup.moshi.Moshi
29 import
30 com.squareup.moshi.kotlin.reflect.KotlinJsonAdapterFa
31 ctory
32 // import kotlinx.coroutines.Deferred
33 import retrofit2.Retrofit
34 import
35 retrofit2.converter.moshi.MoshiConverterFactory
36 import retrofit2.http.GET
37 import retrofit2.http.Query
38
39 private const val BASE_URL =
40 "https://mars.udacity.com/"
41
42 enum class MarsApiFilter(val value: String) {
43     SHOW_RENT("rent"), SHOW_BUY("buy"), SHOW_ALL("all") }

```

```

31
32 /**
33  * Build the Moshi object that Retrofit will be
    using, making sure to add the Kotlin adapter for
34  * full Kotlin compatibility.
35  */
36 private val moshi = Moshi.Builder()
37     .add(KotlinJsonAdapterFactory())
38     .build()
39
40 /**
41  * Use the Retrofit builder to build a retrofit
    object using a Moshi converter with our Moshi
42  * object.
43  */
44 private val retrofit = Retrofit.Builder()
45     .addConverterFactory(MoshiConverterFactory.create(mos
    hi))
46     .baseUrl(BASE_URL)
47     .build()
48
49 /**
50  * A public interface that exposes the
    [getProperties] method
51  */
52 interface MarsApiService {
53     /**
54      * Returns a Coroutine [List] of [MarsProperty]
        which can be fetched with await() if in a Coroutine
        scope.
55      * The @GET annotation indicates that the
        "realestate" endpoint will be requested with the GET
56      * HTTP method
57      */
58     @GET("realestate")
59     suspend fun getProperties(@Query("filter") type:
    String): List<MarsProperty>
60 }
61
62 /**
63  * A public Api object that exposes the lazy-
    initialized Retrofit service

```

64	*/
65	object MarsApi {
66	val retrofitService : MarsApiService by lazy {
	retrofit.create(MarsApiService::class.java) }
67	}

Tabel 14. Source Code Soal 1 Kotlin yang keempat

MarsProperty.kt

1	/*
2	* Copyright 2018, The Android Open Source Project
3	*
4	* Licensed under the Apache License, Version 2.0
	(the "License");
5	* you may not use this file except in compliance
	with the License.
6	* You may obtain a copy of the License at
7	*
8	* http://www.apache.org/licenses/LICENSE-2.0
9	*
10	* Unless required by applicable law or agreed to in
	writing, software
11	* distributed under the License is distributed on
	an "AS IS" BASIS,
12	* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
	either express or implied.
13	* See the License for the specific language
	governing permissions and
14	* limitations under the License.
15	*
16	*/
17	
18	package com.example.android.marsrealestate.network
19	
20	import android.os.Parcelable
21	import androidx.lifecycle.LiveData
22	import
	com.example.android.marsrealestate.overview.MarsApiS
	tatus
23	import com.squareup.moshi.Json
24	import kotlinx.android.parcel.Parcelize
25	
26	/**
27	* Gets Mars real estate property information from

```

the Mars API Retrofit service and updates the
28 * [MarsProperty] and [MarsApiStatus] [LiveData].
The Retrofit service returns a coroutine
29 * Deferred, which we await to get the result of the
transaction.
30 * @param filter the [MarsApiFilter] that is sent as
part of the web server request
31 */
32 @Parcelize
33 data class MarsProperty(
34     val id: String,
35     // used to map img_src from the JSON to
imgSrcUrl in our class
36     @Json(name = "img_src") val imgSrcUrl:
String,
37     val type: String,
38     val price: Double) : Parcelable {
39     val isRental
40     get() = type == "rent"
41 }

```

Tabel 15. Source Code Soal 1 Kotlin yang kelima

BindingAdapters.kt

```

1  /*
2   * Copyright 2018, The Android Open Source Project
3   *
4   * Licensed under the Apache License, Version 2.0
(the "License");
5   * you may not use this file except in compliance
with the License.
6   * You may obtain a copy of the License at
7   *
8   *     http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in
writing, software
11  * distributed under the License is distributed on
an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied.
13  * See the License for the specific language
governing permissions and
14  * limitations under the License.

```



```

15  *
16  */
17
18  package com.example.android.marsrealestate
19
20  import android.view.View
21  import android.widget.ImageView
22  import android.widget.TextView
23  import androidx.core.net.toUri
24  import androidx.databinding.BindingAdapter
25  import androidx.recyclerview.widget.RecyclerView
26  import com.bumptech.glide.Glide
27  import com.bumptech.glide.request.RequestOptions
28  import
    com.example.android.marsrealestate.network.MarsProperty
29  import
    com.example.android.marsrealestate.overview.MarsApiStatus
30  import
    com.example.android.marsrealestate.overview.PhotoGridAdapter
31
32  /**
33   * When there is no Mars property data (data is
    null), hide the [RecyclerView], otherwise show it.
34   */
35  @BindingAdapter("listData")
36  fun bindRecyclerView(recyclerView: RecyclerView,
    data: List<MarsProperty>?) {
37      val adapter = recyclerView.adapter as
    PhotoGridAdapter
38      adapter.submitList(data)
39  }
40
41  /**
42   * Uses the Glide library to load an image by URL
    into an [ImageView]
43   */
44  @BindingAdapter("imageUrl")
45  fun bindImage(imgView: ImageView, imgUrl: String?) {
46      imgUrl?.let {
47          val imgUri =

```

```

imgUrl.toUri().buildUpon().scheme("https").build()
48     Glide.with(imgView.context)
49         .load(imgUri)
50         .apply(RequestOptions())
51     .placeholder(R.drawable.loading_animation)
52     .error(R.drawable.ic_broken_image))
53         .into(imgView)
54     }
55 }
56
57 /**
58  * This binding adapter displays the [MarsApiStatus]
59  * of the network request in an image view. When
60  * the request is loading, it displays a
61  * loading_animation. If the request has an error, it
62  * displays a broken image to reflect the connection
63  * error. When the request is finished, it
64  * hides the image view.
65  */
66 @BindingAdapter("marsApiStatus")
67 fun bindStatus(statusImageView: ImageView, status:
MarsApiStatus?) {
68     when (status) {
69         MarsApiStatus.LOADING -> {
70             statusImageView.visibility =
View.VISIBLE
71
72             statusImageView.setImageResource(R.drawable.loading_
animation)
73         }
74         MarsApiStatus.ERROR -> {
75             statusImageView.visibility =
View.VISIBLE
76
77             statusImageView.setImageResource(R.drawable.ic_conne
ction_error)
78         }
79         MarsApiStatus.DONE -> {
80             statusImageView.visibility = View.GONE
81         }
82     }
83 }

```

78	}
	}

Tabel 16. Source Code Soal 1 Kotlin yang keenam

MainActivity.kt

1	/*
2	* Copyright 2018, The Android Open Source Project
3	*
4	* Licensed under the Apache License, Version 2.0
	(the "License");
5	* you may not use this file except in compliance
	with the License.
6	* You may obtain a copy of the License at
7	*
8	* http://www.apache.org/licenses/LICENSE-2.0
9	*
10	* Unless required by applicable law or agreed to
	in writing, software
11	* distributed under the License is distributed on
	an "AS IS" BASIS,
12	* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
	either express or implied.
13	* See the License for the specific language
	governing permissions and
14	* limitations under the License.
15	*
16	*/
17	
18	package com.example.android.marsrealestate
19	
20	import android.os.Bundle
21	import androidx.appcompat.app.AppCompatActivity
22	
23	class MainActivity : AppCompatActivity() {
24	
25	/**
26	* Our MainActivity is only responsible for
	setting the content view that contains the
27	* Navigation Host.
28	*/
29	override fun onCreate(savedInstanceState:
	Bundle?) {
30	super.onCreate(savedInstanceState)

31	setContentView(R.layout.activity_main)
32	}
33	}

Tabel 17. Source Code Soal 1 Kotlin yang ketujuh

activity_main.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	
3	<!--
4	~ Copyright 2018, The Android Open Source Project
5	~
6	~ Licensed under the Apache License, Version 2.0
7	(the "License");
8	~ you may not use this file except in compliance
9	with the License.
10	~ You may obtain a copy of the License at
11	~
12	~ http://www.apache.org/licenses/LICENSE-2.0
13	~ Unless required by applicable law or agreed to
14	in writing, software
15	~ distributed under the License is distributed on
16	an "AS IS" BASIS,
17	~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
18	either express or implied.
19	~ See the License for the specific language
20	governing permissions and
21	~ limitations under the License.
22	~
23	-->
24	<fragment
25	xmlns:android="http://schemas.android.com/apk/res/a
26	ndroid"
27	xmlns:app="http://schemas.android.com/apk/res-
28	auto"
29	android:id="@+id/nav_host_fragment"
30	android:name="androidx.navigation.fragment.NavHostF
31	ragment"
32	android:layout_width="match_parent"
33	android:layout_height="match_parent"

27	<pre> app:defaultNavHost="true" app:navGraph="@navigation/nav_graph" /> </pre>
----	---

Tabel 18. Source Code Soal 1 XML yang pertama

fragment_detail.xml

1	<code><?xml version="1.0" encoding="utf-8"?></code>
2	
3	<code><!--</code>
4	<code>~ Copyright 2018, The Android Open Source Project</code>
5	<code>~</code>
6	<code>~ Licensed under the Apache License, Version 2.0</code>
	<code>(the "License");</code>
7	<code>~ you may not use this file except in compliance</code>
	<code>with the License.</code>
8	<code>~ You may obtain a copy of the License at</code>
9	<code>~</code>
10	<code>~ http://www.apache.org/licenses/LICENSE-2.0</code>
11	<code>~</code>
12	<code>~ Unless required by applicable law or agreed to</code>
	<code>in writing, software</code>
13	<code>~ distributed under the License is distributed on</code>
	<code>an "AS IS" BASIS,</code>
14	<code>~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,</code>
	<code>either express or implied.</code>
15	<code>~ See the License for the specific language</code>
	<code>governing permissions and</code>
16	<code>~ limitations under the License.</code>
17	<code>~</code>
18	<code>--></code>
19	
20	<code><layout</code>
	<code>xmlns:android="http://schemas.android.com/apk/res/an</code>
	<code>droid"</code>
21	<code>xmlns:app="http://schemas.android.com/apk/res-</code>
	<code>auto"</code>
22	<code>xmlns:tools="http://schemas.android.com/tools"></code>
23	
24	<code><data></code>
25	<code><variable</code>
26	<code>name="viewModel"</code>
27	<code>type="com.example.android.marsrealestate.detail.Deta</code>
	<code>ilViewModel" /></code>

```

28     </data>
29
30     <ScrollView
31         android:layout_width="match_parent"
32         android:layout_height="match_parent"
33         tools:context=".DetailFragment">
34
35     <androidx.constraintlayout.widget.ConstraintLayout
36         android:layout_width="match_parent"
37         android:layout_height="wrap_content"
38         android:padding="16dp">
39
40         <ImageView
41             android:id="@+id/main_photo_image"
42             android:layout_width="0dp"
43             android:layout_height="266dp"
44             android:scaleType="centerCrop"
45
46             app:imageUrl="@{viewModel.selectedProperty.imgSrcUrl}"
47
48             app:layout_constraintEnd_toEndOf="parent"
49             app:layout_constraintStart_toStartOf="parent"
50             app:layout_constraintTop_toTopOf="parent"
51             tools:src="@tools:sample/backgrounds/scenic" />
52
53         <TextView
54             android:id="@+id/property_type_text"
55             android:layout_width="wrap_content"
56             android:layout_height="wrap_content"
57             android:layout_marginTop="16dp"
58
59             android:text="@{viewModel.displayPropertyType}"
60             android:textColor="#de000000"
61             android:textSize="39sp"
62
63             app:layout_constraintStart_toStartOf="parent"
64             app:layout_constraintTop_toBottomOf="@+id/main_photo

```

61	<code>_image"</code>
62	<code>tools:text="To Rent" /></code>
63	<code><TextView</code>
64	<code>android:id="@+id/price_value_text"</code>
65	<code>android:layout_width="wrap_content"</code>
66	<code>android:layout_height="wrap_content"</code>
67	<code>android:layout_marginTop="8dp"</code>
68	<code>android:text="@{viewModel.displayPropertyPrice}"</code>
69	<code>android:textColor="#de000000"</code>
70	<code>android:textSize="20sp"</code>
71	<code>app:layout_constraintStart_toStartOf="parent"</code>
72	<code>app:layout_constraintTop_toBottomOf="@+id/property_t</code>
73	<code>ype_text"</code>
74	<code>tools:text="\$100,000" /></code>
75	<code></androidx.constraintlayout.widget.ConstraintLayout></code>
76	<code></ScrollView></code>
77	<code></layout></code>

Tabel 19. Source Code Soal 1 XML yang kedua

fragment_overview.xml

1	<code><?xml version="1.0" encoding="utf-8"?></code>
2	
3	<code><!--</code>
4	<code>~ Copyright 2018, The Android Open Source Project</code>
5	<code>~</code>
6	<code>~ Licensed under the Apache License, Version 2.0</code>
7	<code>(the "License");</code>
8	<code>~ you may not use this file except in compliance</code>
9	<code>with the License.</code>
10	<code>~ You may obtain a copy of the License at</code>
11	<code>~</code>
12	<code>~ http://www.apache.org/licenses/LICENSE-2.0</code>
13	<code>~ Unless required by applicable law or agreed to</code>
	<code>in writing, software</code>
	<code>~ distributed under the License is distributed on</code>
	<code>an "AS IS" BASIS,</code>

```

14 ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
   either express or implied.
15 ~ See the License for the specific language
   governing permissions and
16 ~ limitations under the License.
17 ~
18 -->
19
20 <layout
   xmlns:android="http://schemas.android.com/apk/res/an
   droid"
21   xmlns:app="http://schemas.android.com/apk/res-
   auto"
22   xmlns:tools="http://schemas.android.com/tools">
23
24   <data>
25       <variable
26           name="viewModel"
27   type="com.example.android.marsrealestate.overview.Ov
   erviewViewModel" />
28   </data>
29
30   <androidx.constraintlayout.widget.ConstraintLayout
31       android:layout_width="match_parent"
32       android:layout_height="match_parent"
33   tools:context="com.example.android.marsrealestate.Ma
   inActivity">
34
35       <androidx.recyclerview.widget.RecyclerView
36           android:id="@+id/photos_grid"
37           android:layout_width="0dp"
38           android:layout_height="0dp"
39           android:padding="6dp"
40           android:clipToPadding="false"
41   app:layoutManager="androidx.recyclerview.widget.Grid
   LayoutManager"
42   app:layout_constraintBottom_toBottomOf="parent"

```



```

43 app:layout_constraintLeft_toLeftOf="parent"
44 app:layout_constraintRight_toRightOf="parent"
45 app:layout_constraintTop_toTopOf="parent"
46     app:listData="@{viewModel.properties}"
47     app:spanCount="2"
48     tools:itemCount="16"
49     tools:listitem="@layout/grid_view_item"
50 />
51     <ImageView
52         android:id="@+id/status_image"
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55 app:layout_constraintBottom_toBottomOf="parent"
56 app:layout_constraintLeft_toLeftOf="parent"
57 app:layout_constraintRight_toRightOf="parent"
58 app:layout_constraintTop_toTopOf="parent"
59     app:marsApiStatus="@{viewModel.status}"
60 />
61 </androidx.constraintlayout.widget.ConstraintLayout>
62 </layout>

```

Tabel 20. Source Code Soal 1 XML yang ketiga

grid_view_item.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2
3  <!--
4      ~ Copyright 2018, The Android Open Source Project
5      ~
6      ~ Licensed under the Apache License, Version 2.0
7      ~ (the "License");
8      ~ you may not use this file except in compliance
9      ~ with the License.
10     ~ You may obtain a copy of the License at
11     ~

```

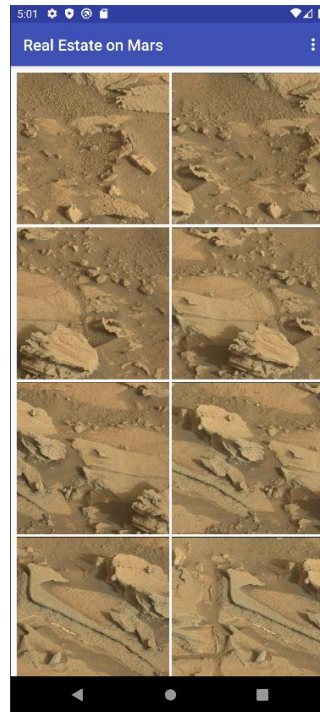
```

10 ~ http://www.apache.org/licenses/LICENSE-2.0
11 ~
12 ~ Unless required by applicable law or agreed to
  in writing, software
13 ~ distributed under the License is distributed on
  an "AS IS" BASIS,
14 ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
  either express or implied.
15 ~ See the License for the specific language
  governing permissions and
16 ~ limitations under the License.
17 ~
18 -->
19
20 <layout
  xmlns:android="http://schemas.android.com/apk/res/an
  droid"
21   xmlns:app="http://schemas.android.com/apk/res-
  auto"
22   xmlns:tools="http://schemas.android.com/tools">
23   <data>
24     <variable
25       name="property"
26 type="com.example.android.marsrealestate.network.Mar
  sProperty" />
27   </data>
28   <ImageView
29     android:id="@+id/mars_image"
30     android:layout_width="match_parent"
31     android:layout_height="170dp"
32     android:scaleType="centerCrop"
33     android:adjustViewBounds="true"
34     android:padding="2dp"
35     app:imageUrl="@{property.imageUrl}"
36 tools:src="@tools:sample/backgrounds/scenic"/>
37 </layout>

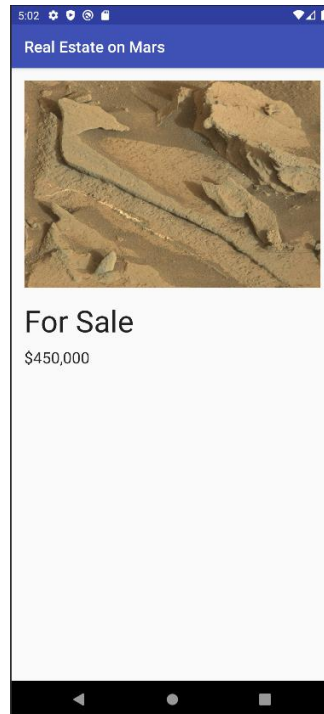
```

Tabel 21. Source Code Soal 1 XML yang keempat

B. Output Program



Gambar 19. Screenshot Hasil Jawaban Soal 1 yang ke 1



Gambar 20. Screenshot Hasil Jawaban Soal 1 yang ke 2

C. Pembahasan

Untuk file DetailFragment.kt

Package dan Imports: Kode dimulai dengan deklarasi `package com.example.android.marsrealestate.detail` dan beberapa import yang diperlukan untuk menggunakan komponen Android dan Android Jetpack, seperti `Fragment`, `ViewModelProvider`, dan `FragmentDetailBinding`.

DetailFragment Class: Kelas ini mewakili sebuah `Fragment` dalam arsitektur aplikasi Android. `Fragment` digunakan untuk menampilkan informasi detail tentang suatu properti real estate Mars.

onCreateView() Method: Method ini dipanggil saat `Fragment` dibuat. Ia meng-inflate layout `Fragment` menggunakan `FragmentDetailBinding`, yang merupakan metode yang disediakan oleh `Data Binding Library` untuk menghubungkan antara layout XML dan kode Kotlin/Java.

LifecycleOwner: Dengan mengatur `lifecycleOwner` dari `binding` menjadi `this`, `Fragment` mengikuti siklus hidup (`lifecycle`) dari `Fragment` tersebut, memastikan bahwa UI akan diperbarui saat perubahan data terjadi.

ViewModel: Fragment ini menggunakan ViewModel (DetailViewModel) untuk memisahkan logika bisnis dari UI. ViewModel ini dipasangkan dengan DetailViewModelFactory yang diinisialisasi dengan marsProperty (properti Mars yang dipilih) dan application.

SafeArgs: Untuk memperoleh selectedProperty yang merupakan properti Mars yang dipilih, Fragment menggunakan SafeArgs dari Jetpack Navigation. SafeArgs memastikan bahwa data yang dilewatkan antar-Fragment aman dan tipe-aman.

Return Statement: Method onCreateView() mengembalikan binding.root, yaitu root View dari layout yang telah di-inflate, yang kemudian akan ditampilkan oleh sistem Android.

Kode ini menunjukkan penggunaan praktik terbaik dalam pengembangan aplikasi Android dengan menggunakan Data Binding, ViewModel, dan SafeArgs untuk mengelola dan menampilkan data dengan cara yang terstruktur dan mudah dikelola.

Untuk file DetailViewModel.kt

Package dan Imports: Kode ini dimulai dengan deklarasi package com.example.android.marsrealestate.detail dan beberapa import yang diperlukan untuk menggunakan komponen Android seperti Application, AndroidViewModel, LiveData, MutableLiveData, dan Transformations.

DetailViewModel Class: Kelas ini merupakan subclass dari AndroidViewModel. Ini berarti ViewModel ini terkait dengan siklus hidup aplikasi Android (Application), memungkinkan ViewModel untuk beroperasi secara aman selama siklus hidup aplikasi.

selectedProperty LiveData: selectedProperty adalah LiveData yang menyimpan informasi tentang properti Mars yang dipilih. MutableLiveData digunakan untuk mengubah nilai properti secara dinamis.

Initialization (init block): Pada saat inisialisasi, _selectedProperty diinisialisasi dengan nilai marsProperty, yang merupakan properti Mars yang dipilih.

Transformations: ViewModel menggunakan Transformations.map untuk mentransformasikan selectedProperty menjadi displayPropertyPrice dan displayPropertyType. Ini memungkinkan ViewModel untuk menyediakan data yang sudah diformat untuk ditampilkan di UI tanpa harus melakukan pemformatan langsung di Fragment.

displayPropertyPrice: LiveData yang diformat untuk menampilkan harga properti sesuai dengan jenisnya (penjualan atau sewa).

displayPropertyType: LiveData yang diformat untuk menampilkan jenis properti (penjualan atau sewa) dalam bentuk string yang sesuai dengan lokal aplikasi.

Application Context: Penggunaan app.applicationContext memungkinkan akses ke konteks aplikasi untuk mengambil string-string lokal yang diperlukan untuk menampilkan informasi properti dengan benar.

Kode ini menunjukkan penggunaan ViewModel untuk mengelola dan memproses data yang diperlukan untuk ditampilkan di UI, dengan memisahkan logika presentasi dari logika bisnis aplikasi. Ini juga memanfaatkan fitur LiveData dan Transformations dari Android Jetpack untuk mengamati dan merespons perubahan data secara efisien.

Untuk file DetailViewModelFactory.kt

Package dan Imports: Kode dimulai dengan deklarasi package com.example.android.marsrealestate.detail dan beberapa import yang diperlukan untuk menggunakan komponen Android seperti Application, ViewModel, ViewModelProvider, dan MarsProperty.

DetailViewModelFactory Class: Kelas ini mengimplementasikan ViewModelProvider.Factory, yang digunakan untuk membuat instance dari DetailViewModel dengan menyediakan marsProperty (properti Mars yang dipilih) dan application (konteks aplikasi Android).

create() Method: Method ini di-override untuk membuat instance dari ViewModel (DetailViewModel). Jika modelClass yang diminta adalah DetailViewModel, maka factory ini akan membuat instance DetailViewModel dengan menggunakan marsProperty dan application yang disediakan.

Type Casting: Terdapat penekanan unchecked_cast karena dalam Kotlin, saat mengkonversi tipe generik, pengecualian tipe dapat dihindari dengan cara ini. Namun, dalam kasus ini, jika kelas yang diminta bukanlah DetailViewModel, akan dilemparkan IllegalArgumentException.

Penggunaan Factory: Factory seperti ini sangat berguna ketika Anda perlu menyediakan parameter tambahan ke dalam ViewModel selain dari konteks aplikasi, seperti objek-objek model seperti marsProperty di sini. Factory memungkinkan pembuatan ViewModel yang lebih fleksibel dan terstruktur dalam aplikasi Android.

Kode ini menunjukkan bagaimana Anda dapat mengorganisir dan memisahkan logika pembuatan ViewModel dari logika ViewModel itu sendiri, memungkinkan aplikasi untuk memenuhi kebutuhan konfigurasi yang berbeda dengan lebih efektif.

Untuk file MarsApiService.kt

Package dan Imports: Kode dimulai dengan deklarasi `package com.example.android.marsrealestate.network` dan beberapa import yang diperlukan untuk menggunakan Retrofit, Moshi, Kotlin adapter, dan beberapa komponen pendukung.

Konstanta dan Enum:

BASE_URL: Menyimpan URL dasar untuk API Mars.

MarsApiFilter: Enum yang mendefinisikan opsi filter untuk properti Mars (SHOW_RENT, SHOW_BUY, SHOW_ALL).

Moshi Configuration:

moshi: Objek Moshi dibangun dengan menggunakan `KotlinJsonAdapterFactory` untuk mendukung serialisasi/deserialisasi dengan Kotlin secara penuh.

Retrofit Configuration:

retrofit: Objek Retrofit dibangun dengan menambahkan converter `MoshiConverterFactory` yang menggunakan objek moshi yang telah dikonfigurasi sebelumnya, serta menentukan BASE_URL dari layanan API.

MarsApiService Interface:

MarsApiService: Interface yang mendefinisikan metode untuk mengambil daftar properti Mars menggunakan Retrofit.

@GET("realestate"): Anotasi `@GET` digunakan untuk menandai bahwa endpoint "realestate" akan diminta dengan metode HTTP GET.

suspend fun getProperties(@Query("filter") type: String): Metode yang mengembalikan daftar properti Mars menggunakan coroutine, dengan parameter `type` yang digunakan untuk filter (rent, buy, atau all).

MarsApi Object:

MarsApi: Objek singleton yang mengekspos layanan Retrofit yang telah diinisialisasi secara malas (lazy-initialized) melalui `retrofit.create(MarsApiService::class.java)`. Singleton ini memastikan bahwa hanya ada satu instance dari layanan Retrofit yang dibuat sepanjang aplikasi.

Kode ini menunjukkan penggunaan Retrofit untuk berkomunikasi dengan API web (dalam hal ini, API properti real estate Mars) dan Moshi untuk mengonversi data JSON ke objek Kotlin. Ini adalah praktik umum dalam pengembangan aplikasi Android untuk

mengambil dan menangani data dari sumber eksternal dengan cara yang efisien dan terstruktur.

Untuk File MarsProperty.kt

Package dan Imports: Kode dimulai dengan deklarasi package `com.example.android.marsrealstate.network` dan beberapa import yang diperlukan untuk menggunakan komponen Kotlin dan Moshi.

MarsProperty Data Class:

`@Parcelize`: Anotasi untuk menandai bahwa kelas ini dapat di-parcelable, yang digunakan dalam konteks aplikasi Android untuk mentransfer objek antar komponen aplikasi, seperti antara Activity atau Fragment.

`data class MarsProperty`: Data class yang merepresentasikan properti real estate Mars.
`val id`: ID properti.

`@Json(name = "img_src") val imgSrcUrl`: Anotasi `@Json` digunakan untuk memetakan field `img_src` dari JSON ke `imgSrcUrl` dalam kelas Kotlin.

`val type`: Jenis properti (misalnya, "rent" untuk sewa).

`val price`: Harga properti dalam tipe data Double.

`val isRental`: Properti komputasi yang mengembalikan true jika jenis properti adalah "rent", dan false jika tidak.

Explanation:

`id`, `imgSrcUrl`, `type`, dan `price` adalah properti-properti dari data class `MarsProperty`, yang merepresentasikan atribut-atribut dari properti real estate Mars.

Anotasi `@Json(name = "img_src")` digunakan untuk mengonversi nama field JSON `img_src` menjadi `imgSrcUrl` dalam kelas Kotlin, memfasilitasi deserialisasi menggunakan Moshi.

`isRental` adalah properti komputasi yang memberikan informasi apakah properti ini untuk sewa atau tidak, berdasarkan nilai `type`.

Data class ini digunakan dalam konteks aplikasi Android untuk mewakili data yang diterima dari layanan Retrofit yang mengambil informasi properti real estate Mars dari API. Ini adalah pendekatan yang umum dalam pengembangan aplikasi Android untuk mengelola dan memproses data yang diterima dari sumber eksternal seperti API web.

Untuk File BindingAdapters.kt

Package dan Imports: Kode dimulai dengan deklarasi package `com.example.android.marsrealestate` dan beberapa import yang diperlukan untuk menggunakan komponen Android seperti `View`, `ImageView`, `TextView`, `RecyclerView`, serta komponen dari Glide dan adapter `PhotoGridAdapter`.

Binding Adapters:

`@BindingAdapter("listData")`: Binding adapter ini digunakan untuk menghubungkan `RecyclerView` dengan data `List<MarsProperty>`. Ketika ada perubahan data, adapter `RecyclerView` (`PhotoGridAdapter`) akan di-update dengan data baru menggunakan `submitList`.

`@BindingAdapter("imageUrl")`: Binding adapter ini digunakan untuk mengambil URL gambar (`imageUrl`) dan memuatnya ke dalam `ImageView` (`imgView`) menggunakan Glide library. Jika `imageUrl` tidak null, maka gambar akan dimuat dari URL yang disediakan dengan opsi placeholder dan error image tertentu.

`@BindingAdapter("marsApiStatus")`: Binding adapter ini menangani tampilan status dari permintaan jaringan (`MarsApiStatus`). Bergantung pada status yang diterima (`LOADING`, `ERROR`, atau `DONE`), `ImageView` (`statusImageView`) akan menampilkan gambar animasi loading, ikon error, atau menyembunyikan `ImageView` sesuai dengan statusnya.

Keterangan Tambahan:

Data Binding Adapters digunakan untuk menghubungkan (bind) data dari `ViewModel` ke tampilan UI tanpa perlu menulis kode boilerplate untuk mengatur tampilan secara manual.

Ini meningkatkan keterbacaan dan pemeliharaan kode karena logika UI terkait dengan data dipisahkan dalam binding adapters.

Kode ini menunjukkan penggunaan praktik terbaik dalam pengembangan aplikasi Android dengan menggunakan Data Binding untuk mengelola tampilan UI secara efisien berdasarkan data yang berasal dari `ViewModel` atau model lainnya.

Untuk File MainActivity.kt

Package dan Imports: Kode dimulai dengan deklarasi package `com.example.android.marsrealestate` dan mengimpor `Bundle` serta `AppCompatActivity` dari pustaka Android.

MainActivity Class:

Kelas ini adalah subclass dari AppCompatActivity, yang merupakan aktivitas utama dari aplikasi. AppCompatActivity adalah bagian dari pustaka Jetpack yang menyediakan fitur-fitur modern dan kompatibilitas dengan perangkat yang lebih lama.

`override fun onCreate(savedInstanceState: Bundle?):` Fungsi ini dipanggil saat aktivitas pertama kali dibuat. Ini adalah tempat yang tepat untuk inisialisasi aktivitas, seperti mengatur tampilan konten dan menginisialisasi komponen yang diperlukan.

`super.onCreate(savedInstanceState):` Memanggil implementasi superclass dari `onCreate` untuk memastikan bahwa aktivitas diinisialisasi dengan benar.

`setContentView(R.layout.activity_main):` Menetapkan tata letak (layout) yang akan digunakan untuk aktivitas ini. Layout yang ditetapkan adalah `activity_main`, yang biasanya berisi tampilan-tampilan UI dan elemen-elemen lainnya yang membentuk antarmuka pengguna untuk aktivitas ini.

Keterangan Tambahan:

MainActivity dalam konteks ini memiliki tanggung jawab yang sederhana, yaitu hanya untuk mengatur tampilan konten yang mengandung Navigation Host. Navigation Host adalah container untuk navigasi fragment dalam aplikasi menggunakan Jetpack Navigation Component.

Aktivitas ini bertindak sebagai wadah bagi fragmen-fragmen lain yang akan ditampilkan sesuai dengan navigasi yang diatur dalam aplikasi. Dengan cara ini, MainActivity menjadi komponen utama yang mengelola navigasi dan transisi antar fragmen.

Kode ini menunjukkan pendekatan yang sederhana dan modular untuk mengelola aktivitas dalam aplikasi Android, di mana MainActivity berfungsi sebagai pengatur tampilan dan navigasi utama, sementara logika spesifik dan UI ditempatkan dalam fragmen yang berbeda.

Untuk file activity_main.xml

XML Structure:

Root Element: `<fragment>` adalah elemen root (akar) dari file XML ini.

Namespaces:

`xmlns:android="http://schemas.android.com/apk/res/android":` Namespace Android untuk atribut dan elemen Android.

`xmlns:app="http://schemas.android.com/apk/res-auto"`: Namespace AppCompatActivity untuk atribut tambahan yang terkait dengan kompatibilitas aplikasi.

Attributes:

`android:id="@+id/nav_host_fragment"`: Menetapkan ID unik untuk fragment ini, yang nantinya dapat digunakan untuk merujuk ke fragment dalam kode Kotlin atau XML lainnya.

`android:name="androidx.navigation.fragment.NavHostFragment"`: Menentukan kelas Java atau Kotlin yang digunakan untuk mengimplementasikan fragment ini, yaitu NavHostFragment dari library AndroidX Navigation Component.

`android:layout_width="match_parent"` dan `android:layout_height="match_parent"`: Menetapkan lebar dan tinggi fragment agar mengisi seluruh ruang yang tersedia dalam parent container.

`app:defaultNavHost="true"`: Menandakan bahwa fragment ini adalah default NavHost untuk aktivitas yang terkait. Ini memungkinkan fragment untuk menangani navigasi antar fragment-fragment lainnya menggunakan Navigation Component.

`app:navGraph="@navigation/nav_graph"`: Menunjukkan file navigasi (`nav_graph.xml`) yang digunakan sebagai konfigurasi navigasi untuk fragment ini. File navigasi ini berisi definisi rute navigasi (destinasi) dan tindakan-tindakan (actions) antar fragment.

Keterangan Tambahan:

Fragment ini berperan penting dalam mengimplementasikan navigasi fragment-to-fragment menggunakan AndroidX Navigation Component.

NavHostFragment bertindak sebagai wadah untuk fragment-fragment lain dalam aplikasi, dan mengelola tumpukan fragment serta perpindahan antar halaman (page).

Dengan menetapkan `app:navGraph` ke sebuah file navigasi (`nav_graph.xml`), fragment ini mengikuti definisi navigasi yang sudah ditetapkan di dalamnya, sehingga memungkinkan navigasi yang terpusat dan mudah dikelola.

File XML ini adalah bagian integral dari penggunaan Navigation Component dalam pengembangan aplikasi Android, yang menyederhanakan pengaturan navigasi dan mempromosikan pola desain yang baik dalam aplikasi yang lebih kompleks.

Untuk File `fragment_detail.xml`

Root Element: <layout> adalah elemen root (akar) dari file XML ini, menandakan bahwa layout ini menggunakan Data Binding.

Namespaces:

xmlns:android="http://schemas.android.com/apk/res/android": Namespace Android untuk atribut dan elemen Android.

xmlns:app="http://schemas.android.com/apk/res-auto": Namespace AppCompatActivity untuk atribut tambahan yang terkait dengan kompatibilitas aplikasi.

xmlns:tools="http://schemas.android.com/tools": Namespace untuk alat bantu pengembangan Android Studio (tools).

Data Binding:

<data>: Bagian ini mendefinisikan variabel yang digunakan dalam Data Binding.

<variable>: Mendefinisikan variabel viewModel dengan tipe DetailViewModel. Variabel ini akan diikat (bind) dengan ViewModel yang terkait dari kelas DetailViewModel.

Scroll View: <ScrollView> adalah container yang memungkinkan tata letaknya bisa di-scroll jika kontennya melebihi ukuran layar.

Constraint Layout:

<ConstraintLayout>: Digunakan sebagai root container di dalam <ScrollView>, mengatur tampilan dengan konstrain untuk memastikan tata letak yang responsif.

Views:

<ImageView>: Menampilkan gambar utama properti dengan menggunakan app:imageUrl yang diikat ke viewModel.selectedProperty.imgSrcUrl. Ini memanfaatkan Data Binding untuk mengatur gambar secara dinamis.

<TextView>:

property_type_text: Menampilkan jenis properti (viewModel.displayPropertyType), yang juga diikat menggunakan Data Binding.

price_value_text: Menampilkan harga properti (viewModel.displayPropertyPrice), juga diikat menggunakan Data Binding.

Tools Namespace:

Digunakan untuk memberikan nilai dummy pada tampilan saat merancang UI di Android Studio (tools:text, tools:src).

Keterangan Tambahan:

Penggunaan Data Binding memungkinkan binding langsung antara model data (ViewModel) dan tampilan (layout XML), mengurangi boilerplate code dan meningkatkan pemeliharaan kode.

Layout ini digunakan dalam konteks DetailFragment, yang berhubungan dengan menampilkan detail properti Mars menggunakan ViewModel yang sesuai (DetailViewModel).

Responsivitas tata letak dijamin oleh penggunaan ConstraintLayout, yang memungkinkan elemen-elemen tampilan untuk diposisikan relatif terhadap elemen-elemen lainnya dengan menggunakan constraints.

File XML ini merupakan bagian integral dari implementasi tampilan detail dalam aplikasi Android, memanfaatkan Data Binding untuk mempermudah pengelolaan dan interaksi antara tampilan dan data aplikasi.

Untuk File fragment_overview.xml

Root Element: <layout> adalah elemen root (akar) dari file XML ini, menandakan bahwa layout ini menggunakan Data Binding.

Namespaces:

xmlns:android="http://schemas.android.com/apk/res/android": Namespace Android untuk atribut dan elemen Android.

xmlns:app="http://schemas.android.com/apk/res-auto": Namespace AppCompatActivity untuk atribut tambahan yang terkait dengan kompatibilitas aplikasi.

xmlns:tools="http://schemas.android.com/tools": Namespace untuk alat bantu pengembangan Android Studio (tools).

Data Binding:

<data>: Bagian ini mendefinisikan variabel yang digunakan dalam Data Binding.

<variable>: Mendefinisikan variabel viewModel dengan tipe OverviewViewModel. Variabel ini akan diikat (bind) dengan ViewModel yang terkait dari kelas OverviewViewModel.

Constraint Layout:

<ConstraintLayout>: Digunakan sebagai root container, mengatur tampilan dengan konstrain untuk memastikan tata letak yang responsif.

Views:

<RecyclerView>: Menampilkan grid foto-foto properti Mars menggunakan app:lista yang diikat ke viewModel.properties. Ini memanfaatkan Data Binding untuk menampilkan daftar properti secara dinamis.

app:layoutManager: Mengatur GridLayoutManager untuk mengatur tata letak grid.

app:spanCount: Menentukan jumlah kolom dalam grid (dalam hal ini, 2 kolom).

tools:itemCount dan tools:listitem: Digunakan oleh Android Studio untuk desain UI di mode desain.

<ImageView>: Menampilkan status dari API Mars menggunakan app:marsApiStatus yang diikat ke viewModel.status. Ini memanfaatkan Data Binding untuk menampilkan status API secara dinamis.

Padding and ClipToPadding:

android:padding="6dp": Memberikan padding sebesar 6dp pada RecyclerView.

android:clipToPadding="false": Mengatur agar padding tidak memengaruhi efek scroll di dalam RecyclerView.

Tools Namespace:

Digunakan untuk memberikan nilai dummy pada tampilan saat merancang UI di Android Studio (tools:itemCount, tools:listitem).

File XML ini digunakan dalam konteks MainActivity atau mungkin juga OverviewFragment, yang terkait dengan menampilkan daftar properti Mars dan status API menggunakan Data Binding untuk mempermudah pengelolaan dan interaksi antara tampilan dan data aplikasi.

Untuk File grid_view_item.xml

Root Element: <layout> adalah elemen root (akar) dari file XML ini, menandakan bahwa layout ini menggunakan Data Binding.

Namespaces:

xmlns:android="http://schemas.android.com/apk/res/android": Namespace Android untuk atribut dan elemen Android.

`xmlns:app="http://schemas.android.com/apk/res-auto"`: Namespace AppCompatActivity untuk atribut tambahan yang terkait dengan kompatibilitas aplikasi.

`xmlns:tools="http://schemas.android.com/tools"`: Namespace untuk alat bantu pengembangan Android Studio (tools).

Data Binding:

`<data>`: Bagian ini mendefinisikan variabel yang digunakan dalam Data Binding.

`<variable>`: Mendefinisikan variabel property dengan tipe `MarsProperty`. Variabel ini akan diikat (bind) dengan objek `MarsProperty` yang digunakan untuk menampilkan gambar properti Mars.

ImageView:

`<ImageView>`: Menampilkan gambar properti Mars dengan mengikat `app:imageUrl` ke `property.imgSrcUrl` menggunakan Data Binding. Ini memungkinkan aplikasi untuk dinamis memuat gambar properti dari URL yang disediakan dalam objek `MarsProperty`.

`android:layout_width="match_parent"` dan `android:layout_height="170dp"`: Menetapkan lebar dan tinggi gambar agar memenuhi lebar layar dan memiliki tinggi tetap.

`android:scaleType="centerCrop"`: Mengatur bagaimana gambar diatur dalam `ImageView` (di sini di-centre dan crop jika perlu).

`android:adjustViewBounds="true"`: Memastikan bahwa gambar disesuaikan dengan batas tampilan `ImageView`.

`tools:src="@tools:sample/backgrounds/scenic"`: Digunakan oleh Android Studio untuk desain UI di mode desain, menampilkan sampel gambar latar belakang alam.

File ini cocok digunakan dalam konteks di mana Anda perlu menampilkan gambar properti Mars dengan menggunakan Data Binding untuk mempermudah pengelolaan dan interaksi antara tampilan dan data aplikasi.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/farlyhaydyhdjalil/Praktikum-PEMROGRAMANMOBILE/tree/4a165f7291f361c80ae284648c0ff6980523075e/Mo%20dul%204/MarsApp>

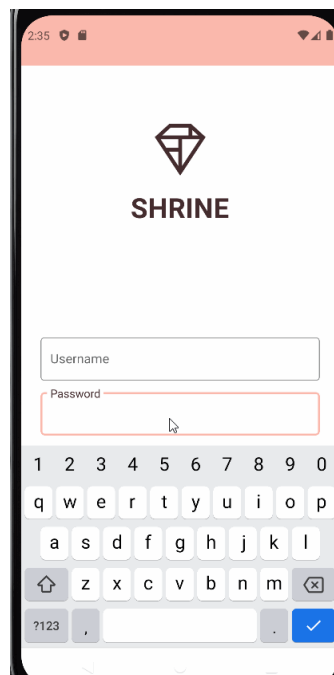
MODUL 5 : ANDROID UI DESIGN

SOAL 1

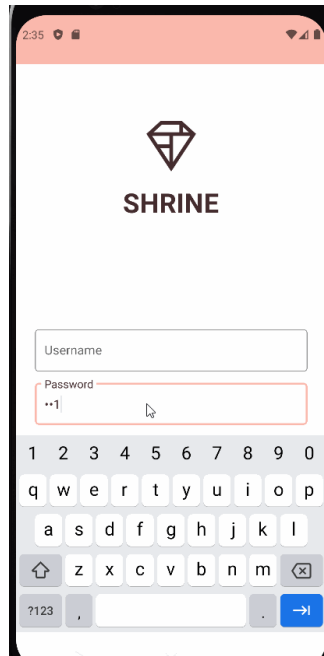
Soal Praktikum:

Buat sebuah aplikasi e-commerce menggunakan Material Design Components (MDC). Aplikasi ini akan menampilkan daftar produk, rincian produk, dan keranjang belanja. User interface untuk halaman utama yang menampilkan daftar produk menggunakan RecyclerView sesuai dengan prinsip Material Design yang akan dibuat. Navigasi antar layar akan ditambahkan menggunakan Navigation Component untuk mengatur navigasi antara halaman utama dan halaman rincian produk. Model data untuk produk, adapter untuk RecyclerView, serta layout untuk item produk juga akan dibuat. Terakhir, ViewModel untuk menyimpan data keranjang belanja akan dibuat dan fungsionalitas untuk menambahkan produk ke keranjang dari halaman rincian produk akan diimplementasikan.

Contoh aplikasi:



Gambar 21. Contoh gambar pertama



Gambar 22. Contoh gambar kedua

A. Source Code

ShrineApplication.kt

```

1 package
  com.example.praktikum.frly.kotlin.shrine.application
2
3 import android.app.Application
4 import androidx.appcompat.app.AppCompatActivity
5
6 class ShrineApplication : Application() {
7     companion object {
8         lateinit var instance: ShrineApplication
9         private set
10    }
11
12    override fun onCreate() {
13        super.onCreate()
14        instance = this
15    }
16
17    AppCompatActivity.setCompatVectorFromResourcesEnabled(true)
18    }

```

18	
19	}

Tabel 22. Source Code Soal 1 Kotlin yang pertama

ImageRequester.kt

1	package
	com.example.praktikum.frly.kotlin.shrine.network
2	
3	import android.content.Context
4	import android.graphics.Bitmap
5	import android.util.LruCache
6	
7	import com.android.volley.RequestQueue
8	import com.android.volley.toolbox.ImageLoader
9	import com.android.volley.toolbox.NetworkImageView
10	import com.android.volley.toolbox.Volley
11	import
	com.example.praktikum.frly.kotlin.shrine.application
	.ShrineApplication
12	
13	/**
14	* Class that handles image requests using Volley.
15	*/
16	object ImageRequester {
17	private val requestQueue: RequestQueue
18	private val imageLoader: ImageLoader
19	private val maxByteSize: Int
20	
21	init {
22	val context = ShrineApplication.instance
23	requestQueue =
	Volley.newRequestQueue(context)
24	requestQueue.start()
25	maxByteSize = calculateMaxByteSize(context)
26	imageLoader = ImageLoader(
27	requestQueue,
28	object : ImageLoader.ImageCache {
29	private val lruCache = object :
	LruCache<String, Bitmap>(maxByteSize) {
30	override fun sizeOf(url:
	String, bitmap: Bitmap): Int {
31	return bitmap.byteCount
32	}

```

33         }
34
35         @Synchronized
36         override fun getBitmap(url:
String): Bitmap? {
37             return lruCache.get(url)
38         }
39
40         @Synchronized
41         override fun putBitmap(url:
String, bitmap: Bitmap) {
42             lruCache.put(url, bitmap)
43         }
44     })
45 }
46
47 /**
48  * Sets the image on the given
49  * [NetworkImageView] to the image at the given URL
50  * @param networkImageView [NetworkImageView] to
set image on
51  * @param url              URL of the image
52  */
53 fun setImageFromUrl(networkImageView:
54 NetworkImageView, url: String) {
55     networkImageView.setImageUrl(url,
imageLoader)
56 }
57 private fun calculateMaxByteSize(context:
Context): Int {
58     val displayMetrics =
context.resources.displayMetrics
59     val screenBytes = displayMetrics.widthPixels
* displayMetrics.heightPixels * 4
60     return screenBytes * 3
61 }
62 }

```

Tabel 23. Source Code Soal 1 Kotlin yang kedua

ProductEntry.kt

1	package
	com.example.praktikum.frly.kotlin.shrine.network
2	
3	import android.content.res.Resources
4	import android.net.Uri
5	import com.example.praktikum.frly.kotlin.shrine.R
6	import com.google.gson.Gson
7	import com.google.gson.reflect.TypeToken
8	import java.io.BufferedReader
9	import java.util.*
10	
11	<i>/**</i>
12	<i> * A product entry in the list of products.</i>
13	<i> */</i>
14	class ProductEntry(15 val title: String, dynamicUrl: String, val url: String, val price: String, val description: String) { 16 val dynamicUrl: Uri = Uri.parse(dynamicUrl) 17 18 companion object { 19 <i>/**</i> 20 <i> * Loads a raw JSON at R.raw.products and</i> <i>converts it into a list of ProductEntry objects</i> 21 <i> */</i> 22 fun initProductEntryList(resources: Resources): List<ProductEntry> { 23 val inputStream = resources.openRawResource(R.raw.products) 24 val jsonProductsString = inputStream.bufferedReader().use(BufferedReader::rea dText) 25 val gson = Gson() 26 val productListType = object : TypeToken<ArrayList<ProductEntry>>() {}.type 27 return gson.fromJson<List<ProductEntry>>(jsonProductsString , productListType) 28 } 29 } 30 }

Tabel 24. Source Code Soal 1 Kotlin yang ketiga

StaggeredProductCardRecyclerViewAdapter.kt

```
1 package
  com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6
7 import com.example.praktikum.frly.kotlin.shrine.R
8 import
  com.example.praktikum.frly.kotlin.shrine.network.ImageRequester
9 import
  com.example.praktikum.frly.kotlin.shrine.network.ProductEntry
10
11 /**
12  * Adapter used to show an asymmetric grid of
  products, with 2 items in the first column, and 1
13  * item in the second column, and so on.
14  */
15 class
  StaggeredProductCardRecyclerViewAdapter(private val
  productList: List<ProductEntry>?) :
  RecyclerView.Adapter<StaggeredProductCardViewHolder>
  () {
16
17     override fun getItemViewType(position: Int): Int
  {
18         return position % 3
19     }
20
21     override fun onCreateViewHolder(parent:
  ViewGroup, viewType: Int):
  StaggeredProductCardViewHolder {
22         var layoutId =
  R.layout.shr_staggered_product_card_first
23         if (viewType == 1) {
24             layoutId =
  R.layout.shr_staggered_product_card_second
25         } else if (viewType == 2) {
```

26	layoutId =
27	R.layout.shr_staggered_product_card_third
28	}
29	val layoutView =
	LayoutInflater.from(parent.context).inflate(layoutId
	, parent, false)
30	return
	StaggeredProductCardViewHolder(layoutView)
31	}
32	
33	override fun onBindViewHolder(holder:
	StaggeredProductCardViewHolder, position: Int) {
34	if (productList != null && position <
	productList.size) {
35	val product = productList[position]
36	holder.productTitle.text = product.title
37	holder.productPrice.text = product.price
38	ImageRequester.setImageFromUrl(holder.productImage,
	product.url)
39	}
40	}
41	
42	override fun getItemCount(): Int {
43	return productList?.size ?: 0
44	}
45	}

Tabel 25. Source Code Soal 1 Kotlin yang keempat

StaggeredProductCardViewHolder.kt

1	package
	com.example.praktikum.frly.kotlin.shrine.staggeredgr
	idlayout
2	
3	import android.view.View
4	import android.widget.TextView
5	import androidx.recyclerview.widget.RecyclerView
6	
7	import com.android.volley.toolbox.NetworkImageView
8	import com.example.praktikum.frly.kotlin.shrine.R

9	
10	class StaggeredProductCardViewHolder(itemView: View)
	: RecyclerView.ViewHolder(itemView) {
11	
12	var productImage: NetworkImageView =
	itemView.findViewById(R.id.product_image)
13	var productTitle: TextView =
	itemView.findViewById(R.id.product_title)
14	var productPrice: TextView =
	itemView.findViewById(R.id.product_price)
15	}

Tabel 26. Source Code Soal 1 Kotlin yang kelima

LoginFragment.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import android.os.Bundle
4	import android.text.Editable
5	import android.view.LayoutInflater
6	import android.view.View
7	import android.view.ViewGroup
8	import android.widget.Button
9	import android.widget.EditText
10	import androidx.fragment.app.Fragment
11	import
	com.google.android.material.textfield.TextInputLayout
	t
12	import com.example.praktikum.frly.kotlin.shrine.R
13	
14	/**
15	* Fragment representing the login screen for
	Shrine.
16	*/
17	class LoginFragment : Fragment() {
18	
19	override fun onCreateView(
20	inflater: LayoutInflater, container:
	ViewGroup?, savedInstanceState: Bundle?): View? {
21	// Inflate the layout for this fragment
22	val view =
	inflater.inflate(R.layout.shr_login_fragment,
	container, false)
23	

```

24         // Set an error if the password is less than
        8 characters.

25 view.findViewById<Button>(R.id.next_button).setOnCli
    ckListener {
26         if
        (!isPasswordValid(view.findViewById<EditText>(R.id.p
            assword_edit_text).text)) {

27 view.findViewById<TextInputLayout>(R.id.password_tex
    t_input).error =
        getString(R.string.shr_error_password)
28         } else {

29 view.findViewById<TextInputLayout>(R.id.password_tex
    t_input).error = null // Clear the error
30         (activity as
        NavigationHost).navigateTo(ProductGridFragment(),
            false) // Navigate to the next Fragment
31         }
32     }
33
34         // Clear the error once more than 8
        characters are typed.

35 view.findViewById<EditText>(R.id.password_edit_text)
    .setOnKeyListener { _, _, _ ->
36         if
        (isPasswordValid(view.findViewById<EditText>(R.id.pa
            ssword_edit_text).text)) {

37 view.findViewById<TextInputLayout>(R.id.password_tex
    t_input).error = null //Clear the error
38         }
39         false
40     }
41     return view
42 }
43
44 /*
45     In reality, this will have more complex
        logic including, but not limited to, actual
46     authentication of the username and password.

```


47	<code>*/</code>
48	<code>private fun isPasswordValid(text: Editable?):</code>
49	<code>Boolean {</code>
	<code>return text != null && text.length >= 8</code>
50	<code>}</code>
51	<code>}</code>

Tabel 27. Source Code Soal 1 Kotlin yang keenam

MainActivity.kt

1	<code>package com.example.praktikum.frly.kotlin.shrine</code>
2	
3	<code>import android.os.Bundle</code>
4	<code>import androidx.appcompat.app.AppCompatActivity</code>
5	<code>import androidx.fragment.app.Fragment</code>
6	<code>import com.example.praktikum.frly.kotlin.shrine.R</code>
7	
8	<code>class MainActivity : AppCompatActivity(),</code>
	<code>NavigationHost {</code>
9	
10	<code>override fun onCreate(savedInstanceState:</code>
	<code>Bundle?) {</code>
11	<code>super.onCreate(savedInstanceState)</code>
12	<code>setContentView(R.layout.shr_main_activity)</code>
13	
14	<code>if (savedInstanceState == null) {</code>
15	<code>supportFragmentManager</code>
16	<code>.beginTransaction()</code>
17	<code>.add(R.id.container,</code>
	<code>LoginFragment())</code>
18	<code>.commit()</code>
19	<code>}</code>
20	<code>}</code>
21	
22	<code>/**</code>
23	<code>* Navigate to the given fragment.</code>
24	<code>* </code>
25	<code>* @param fragment Fragment to navigate</code>
	<code>to.</code>
26	<code>* @param addToBackStack Whether or not the</code>
	<code>current fragment should be added to the backstack.</code>
27	<code>*/</code>
28	<code>override fun navigateTo(fragment: Fragment,</code>
	<code>addToBackStack: Boolean) {</code>

29	val transaction = supportFragmentManager
30	.beginTransaction()
31	.replace(R.id.container, fragment)
32	
33	if (addToBackStack) {
34	transaction.addToBackStack(null)
35	}
36	
37	transaction.commit()
38	}
39	}

Tabel 28. Source Code Soal 1 Kotlin yang ketujuh

NavigationHost.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import androidx.fragment.app.Fragment
4	
5	
6	/**
7	* A host (typically an `Activity`) that can
8	display fragments and knows how to respond to
9	* navigation events.
10	*/
11	interface NavigationHost {
12	/**
13	* Trigger a navigation to the specified
14	fragment, optionally adding a transaction to the
15	back
16	* stack to make this navigation reversible.
17	*/
18	fun navigateTo(fragment: Fragment,
19	addToBackStack: Boolean)
20	}

Tabel 29. Source Code Soal 1 Kotlin yang kedelapan

NavigationIconClickListener.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import android.animation.AnimatorSet
4	import android.animation.ObjectAnimator
5	import android.app.Activity
6	import android.content.Context

```

7 import android.graphics.drawable.Drawable
8 import android.util.DisplayMetrics
9 import android.view.View
10 import android.view.animation.Interpolator
11 import android.widget.ImageView
12 import com.example.praktikum.frly.kotlin.shrine.R
13
14 /**
15  * [android.view.View.OnClickListener] used to
16  * the Y-axis when the navigation icon in the toolbar
17  * is pressed.
18  */
19 class NavigationIconClickListener @JvmOverloads
20     internal constructor(
21         private val context: Context, private val
22         sheet: View, private val interpolator: Interpolator?
23         = null,
24         private val openIcon: Drawable? = null,
25         private val closeIcon: Drawable? = null) :
26     View.OnClickListener {
27
28     private val animatorSet = AnimatorSet()
29     private val height: Int
30     private var backdropShown = false
31
32     init {
33         val displayMetrics = DisplayMetrics()
34         (context as
35         Activity).windowManager.defaultDisplay.getMetrics(displayMetrics)
36         height = displayMetrics.heightPixels
37     }
38
39     override fun onClick(view: View) {
40         backdropShown = !backdropShown
41
42         // Cancel the existing animations
43         animatorSet.removeAllListeners()
44         animatorSet.end()
45         animatorSet.cancel()
46
47         updateIcon(view)

```

```

41
42         val translateY = height -
context.resources.getDimensionPixelSize(R.dimen.shr_p
roduct_grid_reveal_height)
43
44         val animator = ObjectAnimator.ofFloat(sheet,
"translationY", (if (backdropShown) translateY else
0).toFloat())
45         animator.duration = 500
46         if (interpolator != null) {
47             animator.interpolator = interpolator
48         }
40         animatorSet.play(animator)
50         animator.start()
51     }
52
53     private fun updateIcon(view: View) {
54         if (openIcon != null && closeIcon != null) {
55             if (view !is ImageView) {
56                 throw
IllegalArgumentException("updateIcon() must be called
on an ImageView")
57             }
58             if (backdropShown) {
59                 view.setImageDrawable(closeIcon)
60             } else {
61                 view.setImageDrawable(openIcon)
62             }
63         }
64     }
65 }

```

Tabel 30. Source Code Soal 1 Kotlin yang kesembilan

ProductCardRecyclerViewAdapter.kt

```

1 package com.example.praktikum.frly.kotlin.shrine
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.praktikum.frly.kotlin.shrine.R
7 import
com.example.praktikum.frly.kotlin.shrine.network.Ima
geRequester

```

8	
9	import
	com.example.praktikum.frly.kotlin.shrine.network.ProductEntry
10	
11	/**
12	* Adapter used to show a simple grid of products.
13	*/
14	class ProductCardRecyclerViewAdapter internal
	constructor(private val productList:
	List<ProductEntry>) :
	RecyclerView.Adapter<ProductCardViewHolder>() {
15	
16	override fun onCreateViewHolder(parent:
	ViewGroup, viewType: Int): ProductCardViewHolder {
17	val layoutView =
	LayoutInflater.from(parent.context).inflate(R.layout
	.shr_product_card, parent, false)
18	return ProductCardViewHolder(layoutView)
19	}
20	
21	override fun onBindViewHolder(holder:
	ProductCardViewHolder, position: Int) {
22	if (position < productList.size) {
23	val product = productList[position]
24	holder.productTitle.text = product.title
25	holder.productPrice.text = product.price
26	ImageRequester.setImageFromUrl(holder.productImage,
	product.url)
27	}
28	}
29	
30	override fun getItemCount(): Int {
31	return productList.size
32	}
33	}

Tabel 31. Source Code Soal 1 Kotlin yang kesepuluh

ProductCardViewHolder.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import android.view.View

5	import android.widget.TextView
6	import androidx.recyclerview.widget.RecyclerView
7	
8	import com.android.volley.toolbox.NetworkImageView
9	import com.example.praktikum.frly.kotlin.shrine.R
10	class ProductCardViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
11	
12	var productImage: NetworkImageView = itemView.findViewById(R.id.product_image)
13	var productTitle: TextView = itemView.findViewById(R.id.product_title)
14	var productPrice: TextView = itemView.findViewById(R.id.product_price)
15	}

Tabel 32. Source Code Soal 1 Kotlin yang kesebelas

ProductGridFragment.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import android.os.Bundle
5	import android.view.LayoutInflater
6	import android.view.Menu
7	import android.view.MenuInflater
8	import android.view.View
9	import android.view.ViewGroup
10	import android.view.animation.AccelerateDecelerateInterpolator
11	import androidx.appcompat.app.AppCompatActivity
12	import androidx.appcompat.widget.Toolbar
13	import androidx.core.content.ContextCompat
14	import androidx.core.widget.NestedScrollView
15	import androidx.fragment.app.Fragment
16	import androidx.recyclerview.widget.GridLayoutManager
17	import androidx.recyclerview.widget.RecyclerView
18	import com.example.praktikum.frly.kotlin.shrine.R
19	import com.example.praktikum.frly.kotlin.shrine.network.ProductEntry
20	import com.example.praktikum.frly.kotlin.shrine.staggeredgrid

```

dlayout.StaggeredProductCardRecyclerViewAdapter

21 class ProductGridFragment : Fragment() {
22
23     override fun onCreate(savedInstanceState:
Bundle?) {
24         super.onCreate(savedInstanceState)
25         setHasOptionsMenu(true)
26     }
27
28     override fun onCreateView(
29         inflater: LayoutInflater, container:
ViewGroup?, savedInstanceState: Bundle?): View? {
30         // Inflate the layout for this fragment with
the ProductGrid theme
31         val view =
inflater.inflate(R.layout.shr_product_grid_fragment,
container, false)
32
33         val app_bar =
view.findViewById<Toolbar>(R.id.app_bar)
34         val recycler_view =
view.findViewById<RecyclerView>(R.id.recycler_view)
35         val product_grid =
view.findViewById<NestedScrollView>(R.id.product_grid
)
36
37         // Set up the tool bar
38         (activity as
AppCompatActivity).setSupportActionBar(app_bar)
39         app_bar.setNavigationOnClickListener(
40             NavigationIconClickListener(
41                 requireActivity(),
42                 product_grid,
43                 AccelerateDecelerateInterpolator(),
44                 ContextCompat.getDrawable(requireContext(),
R.drawable.shr_branded_menu), // Menu open icon
45                 ContextCompat.getDrawable(requireContext(),
R.drawable.shr_close_menu))
46             ) // Menu close icon
47

```

```

48         // Set up the RecyclerView
49         recycler_view.setHasFixedSize(true)
50         val layoutManager =
51     GridLayoutManager(context, 2,
        RecyclerView.HORIZONTAL, false)
        layoutManager.spanSizeLookup = object :
        GridLayoutManager.SpanSizeLookup() {
52             override fun getSpanSize(position: Int):
        Int {
53                 return if (position % 3 == 2) 2 else
        1
54             }
55         }
56         recycler_view.layoutManager =
        layoutManager
57         val adapter =
        StaggeredProductCardRecyclerViewAdapter(
58     ProductEntry.initProductEntryList(resources))
59         recycler_view.adapter = adapter
60         val largePadding =
        resources.getDimensionPixelSize(R.dimen.shr_staggered
        _product_grid_spacing_large)
61         val smallPadding =
        resources.getDimensionPixelSize(R.dimen.shr_staggered
        _product_grid_spacing_small)
62     recycler_view.addItemDecoration(ProductGridItemDecora
        tion(largePadding, smallPadding))
63
64         // Set cut corner background for API 23+
65         product_grid.background =
        context?.getDrawable(R.drawable.shr_product_grid_back
        ground_shape)
66
67         return view
68     }
69
70     override fun onCreateOptionsMenu(menu: Menu,
        menuInflater: MenuInflater) {
71         menuInflater.inflate(R.menu.shr_toolbar_menu,
        menu)
72         super.onCreateOptionsMenu(menu, menuInflater)

```


73	}
74	}

Tabel 33. Source Code Soal 1 Kotlin yang kedua belas

ProductGridItemDecoration.kt

1	package com.example.praktikum.frly.kotlin.shrine
2	
3	import android.graphics.Rect
5	import android.view.View
6	import androidx.recyclerview.widget.RecyclerView
7	/**
8	* Custom item decoration for a vertical
9	[ProductGridFragment] [RecyclerView]. Adds a
10	* small amount of padding to the left of grid
11	items, and a large amount of padding to the right.
12	*/
13	class ProductGridItemDecoration(private val
14	largePadding: Int, private val smallPadding: Int) :
15	RecyclerView.ItemDecoration() {
16	
17	override fun getItemOffsets(outRect: Rect,
18	view: View,
19	
20	parent:
21	RecyclerView, state: RecyclerView.State) {
22	outRect.left = smallPadding
23	outRect.right = largePadding
24	}
25	}

Tabel 34. Source Code Soal 1 Kotlin yang ketiga belas

shr_backdrop.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<merge
3	xmlns:android="http://schemas.android.com/apk/res/a
4	ndroid">
5	
6	<com.google.android.material.button.MaterialButton
7	
8	style="@style/Widget.Shrine.Button.TextButton"
9	android:layout_width="wrap_content"
10	android:layout_height="wrap_content"

8	android:text="@string/shr_featured_label"
9	</>
10	<com.google.android.material.button.MaterialButton
11	style="@style/Widget.Shrine.Button.TextButton"
12	android:layout_width="wrap_content"
13	android:layout_height="wrap_content"
14	android:text="@string/shr_apartment_label"
15	</>
16	<com.google.android.material.button.MaterialButton
17	style="@style/Widget.Shrine.Button.TextButton"
18	android:layout_width="wrap_content"
19	android:layout_height="wrap_content"
20	android:text="@string/shr_accessories_label" />
21	
22	<com.google.android.material.button.MaterialButton
23	style="@style/Widget.Shrine.Button.TextButton"
24	android:layout_width="wrap_content"
25	android:layout_height="wrap_content"
26	android:text="@string/shr_shoes_label" />
27	
28	<com.google.android.material.button.MaterialButton
29	style="@style/Widget.Shrine.Button.TextButton"
30	android:layout_width="wrap_content"
31	android:layout_height="wrap_content"
32	android:text="@string/shr_tops_label" />
33	
34	<com.google.android.material.button.MaterialButton
35	style="@style/Widget.Shrine.Button.TextButton"
36	android:layout_width="wrap_content"
37	android:layout_height="wrap_content"

38	android:text="@string/shr_bottoms_label" />
39	
40	<com.google.android.material.button.MaterialButton
41	style="@style/Widget.Shrine.Button.TextButton"
42	android:layout_width="wrap_content"
43	android:layout_height="wrap_content"
44	android:text="@string/shr_dresses_label" />
45	
46	<View
47	android:layout_width="56dp"
48	android:layout_height="1dp"
49	android:layout_margin="16dp"
50	android:background="?android:attr/textColorPrimary"
	/>
51	
52	<com.google.android.material.button.MaterialButton
53	style="@style/Widget.Shrine.Button.TextButton"
54	android:layout_width="wrap_content"
55	android:layout_height="wrap_content"
56	android:text="@string/shr_account_label" />
57	
58	</merge>

Tabel 35. Source Code Soal 1 XML yang pertama

shr_login_fragment.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<ScrollView
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	xmlns:tools="http://schemas.android.com/tools"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	android:background="@color/loginPageBackgroundColor"
8	tools:context="com.example.praktikum.frly.kotlin.shr

```

9      ine.LoginFragment">
10      <LinearLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:clipChildren="false"
14          android:clipToPadding="false"
15          android:orientation="vertical"
16          android:padding="24dp"
17          android:paddingTop="16dp">
18
19          <ImageView
20              android:layout_width="64dp"
21              android:layout_height="64dp"
22          android:layout_gravity="center_horizontal"
23              android:layout_marginTop="48dp"
24              android:layout_marginBottom="16dp"
25              app:srcCompat="@drawable/shr_logo"
26          android:contentDescription="@string/shr_logo_content
27              _description"
28
29          app:tint="?android:attr/textColorPrimary" />
30
31          <TextView
32              android:layout_width="wrap_content"
33              android:layout_height="wrap_content"
34          android:layout_gravity="center_horizontal"
35              android:layout_marginBottom="132dp"
36              android:text="@string/shr_app_name"
37          android:textAppearance="@style/TextAppearance.Shrine
38              .Title" />
39
40          <com.google.android.material.textfield.TextInputLayout
41              style="@style/Widget.Shrine.TextInputLayout"
42              android:layout_width="match_parent"
43              android:layout_height="wrap_content"

```

```

41 android:hint="@string/shr_hint_username">
42
43 <com.google.android.material.textfield.TextInputEdit
44 Text
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:inputType="text"
48         android:maxLines="1" />
49
50 </com.google.android.material.textfield.TextInputLayout>
51
52 <com.google.android.material.textfield.TextInputLayout
53 ut
54         android:id="@+id/password_text_input"
55
56 style="@style/Widget.Shrine.TextInputLayout"
57         android:layout_width="match_parent"
58         android:layout_height="wrap_content"
59         android:hint="@string/shr_hint_password"
60         app:errorEnabled="true">
61
62 <com.google.android.material.textfield.TextInputEditText
63 Text
64         android:id="@+id/password_edit_text"
65         android:layout_width="match_parent"
66         android:layout_height="wrap_content"
67         android:inputType="textPassword" />
68
69 </com.google.android.material.textfield.TextInputLayout>
70
71 <RelativeLayout
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content">
74
75 <com.google.android.material.button.MaterialButton
76         android:id="@+id/next_button"

```

```

71         style="@style/Widget.Shrine.Button"
72         android:layout_width="wrap_content"
73         android:layout_height="wrap_content"
74         android:layout_alignParentEnd="true"
75 android:layout_alignParentRight="true"
76 android:text="@string/shr_button_next" />
77
78 <com.google.android.material.button.MaterialButton
79         android:id="@+id/cancel_button"
80 style="@style/Widget.Shrine.Button.TextButton"
81         android:layout_width="wrap_content"
82         android:layout_height="wrap_content"
83         android:layout_marginEnd="12dp"
84         android:layout_marginRight="12dp"
85 android:layout_toStartOf="@id/next_button"
86 android:layout_toLeftOf="@id/next_button"
87 android:text="@string/shr_button_cancel" />
88
89     </RelativeLayout>
90 </LinearLayout>
91 </ScrollView>

```

Tabel 36. Source Code Soal 1 XML yang kedua

shr_main_activity.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:id="@+id/container"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8
9     tools:context="com.example.praktikum.frly.kotlin.shrine.MainActivity" />

```

Tabel 37. Source Code Soal 1 XML yang ketiga

shr_product_card.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <com.google.android.material.card.MaterialCardView
   xmlns:android="http://schemas.android.com/apk/res/an
   droid"
3      xmlns:app="http://schemas.android.com/apk/res-
   auto"
4      android:layout_width="match_parent"
5      android:layout_height="wrap_content"
6
7      app:cardBackgroundColor="@android:color/transparent"
8      app:cardElevation="0dp"
9      app:cardPreventCornerOverlap="true">
10
11      <LinearLayout
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:orientation="vertical">
15
16          <com.android.volley.toolbox.NetworkImageView
17              android:id="@+id/product_image"
18              android:layout_width="match_parent"
19              android:layout_height="wrap_content"
20
21              android:background="?attr/colorPrimaryDark"
22              android:scaleType="centerCrop" />
23
24          <LinearLayout
25              android:layout_width="match_parent"
26              android:layout_height="wrap_content"
27              android:orientation="vertical"
28              android:padding="16dp">
29
30              <TextView
31                  android:id="@+id/product_title"
32                  android:layout_width="match_parent"
33                  android:layout_height="wrap_content"
34
35                  android:text="@string/shr_product_title"
36                  android:textAlignment="center"
37
38                  android:textAppearance="?attr/textAppearanceSubtitle
```

	2" />
35	
36	<TextView
37	android:id="@+id/product_price"
38	android:layout_width="match_parent"
39	android:layout_height="wrap_content"
40	android:text="@string/shr_product_description"
41	android:textAlignment="center"
42	android:textAppearance="?attr/textAppearanceBody2"
	/>
43	</LinearLayout>
44	</LinearLayout>
45	</com.google.android.material.card.MaterialCardView>

Tabel 38. Source Code Soal 1 XML yang keempat

shr_product_grid_fragment.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<FrameLayout
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	xmlns:tools="http://schemas.android.com/tools"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	android:clipChildren="false"
8	android:clipToPadding="false"
9	tools:context="com.example.praktikum.frly.kotlin.shrine.ProductGridFragment">
10	
11	<LinearLayout
12	style="@style/Widget.Shrine.Backdrop"
13	android:layout_width="match_parent"
14	android:layout_height="match_parent"
15	android:gravity="center_horizontal"
16	android:orientation="vertical"
17	android:paddingTop="88dp">
18	
19	<include layout="@layout/shr_backdrop" />
20	</LinearLayout>


```

21
22     <com.google.android.material.appbar.AppBarLayout
23         android:layout_width="match_parent"
24         android:layout_height="wrap_content"
25         app:elevation="0dp">
26
27         <androidx.appcompat.widget.Toolbar
28             android:id="@+id/app_bar"
29             style="@style/Widget.Shrine.Toolbar"
30             android:layout_width="match_parent"
31             android:layout_height="?attr/actionBarSize"
32             android:paddingStart="12dp"
33             android:paddingLeft="12dp"
34             android:paddingEnd="12dp"
35             android:paddingRight="12dp"
36             app:contentInsetStart="0dp"
37             app:navigationIcon="@drawable/shr_branded_menu"
38             app:title="@string/shr_app_name" />
39     </com.google.android.material.appbar.AppBarLayout>
40
41     <androidx.core.widget.NestedScrollView
42         android:id="@+id/product_grid"
43         android:layout_width="match_parent"
44         android:layout_height="match_parent"
45         android:layout_marginTop="56dp"
46         android:background="@color/productGridBackgroundColor"
47         android:elevation="8dp"
48         app:layout_behavior="@string/appbar_scrolling_view_behavior">
49
50         <androidx.recyclerview.widget.RecyclerView
51             android:id="@+id/recycler_view"
52             android:layout_width="match_parent"
53             android:layout_height="match_parent" />
54
55     </androidx.core.widget.NestedScrollView>

```

56	
57	</FrameLayout>

Tabel 39. Source Code Soal 1 XML yang kelima

shr_staggered_product_card_first.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<com.google.android.material.card.MaterialCardView
	xmlns:android="http://schemas.android.com/apk/res/and
	roid"
3	xmlns:app="http://schemas.android.com/apk/res-
	auto"
4	android:layout_width="wrap_content"
5	android:layout_height="match_parent"
6	app:cardBackgroundColor="@android:color/transparent"
7	app:cardElevation="0dp"
8	app:cardPreventCornerOverlap="true"
9	app:contentPaddingTop="@dimen/shr_staggered_product_g
	rid_margin_top_first">
10	
11	<!-- Product card layout used in MDC-103 and
	onward -->
12	<LinearLayout
13	android:layout_width="@dimen/shr_staggered_product_gr
	id_card_width_landscape"
14	android:layout_height="wrap_content"
15	android:layout_marginStart="@dimen/shr_staggered_produ
	ct_grid_spacing_small"
16	android:layout_marginLeft="@dimen/shr_staggered_produ
	ct_grid_spacing_small"
17	android:orientation="vertical">
18	
19	<com.android.volley.toolbox.NetworkImageView
20	android:id="@+id/product_image"
21	android:layout_width="match_parent"
22	android:layout_height="@dimen/shr_staggered_product_g
	rid_card_height_landscape"

23	android:background="?attr/colorPrimaryDark"
24	android:scaleType="centerCrop" />
25	
26	<LinearLayout
27	android:layout_width="match_parent"
28	android:layout_height="wrap_content"
29	android:orientation="vertical"
30	android:padding="16dp">
31	
32	<TextView
33	android:id="@+id/product_title"
34	android:layout_width="match_parent"
35	android:layout_height="wrap_content"
36	android:text="@string/shr_product_title"
37	android:textAlignment="center"
38	android:textAppearance="?attr/textAppearanceSubtitle2"
39	/>
40	<TextView
41	android:id="@+id/product_price"
42	android:layout_width="match_parent"
43	android:layout_height="wrap_content"
44	android:text="@string/shr_product_description"
45	android:textAlignment="center"
46	android:textAppearance="?attr/textAppearanceBody2" />
47	</LinearLayout>
48	</LinearLayout>
49	</com.google.android.material.card.MaterialCardView>

Tabel 40. Source Code Soal 1 XML yang keenam

shr_staggered_product_card_second.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<com.google.android.material.card.MaterialCardView
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	android:layout_width="wrap_content"
5	android:layout_height="match_parent"

```

6 app:cardBackgroundColor="@android:color/transparent"
7   app:cardElevation="0dp"
8   app:cardPreventCornerOverlap="true"
9 app:contentPaddingTop="@dimen/shr_staggered_product_g
rid_margin_top_second">
10
11   <!-- Product card layout used in MDC-103 and
onward -->
12   <LinearLayout
13 android:layout_width="@dimen/shr_staggered_product_gr
id_card_width_landscape"
14     android:layout_height="wrap_content"
15 android:layout_marginEnd="@dimen/shr_staggered_produc
t_grid_spacing_small"
16 android:layout_marginRight="@dimen/shr_staggered_prod
uct_grid_spacing_small"
17     android:orientation="vertical">
18
19     <com.android.volley.toolbox.NetworkImageView
20         android:id="@+id/product_image"
21         android:layout_width="match_parent"
22 android:layout_height="@dimen/shr_staggered_product_g
rid_card_height_landscape"
23 android:background="?attr/colorPrimaryDark"
24         android:scaleType="centerCrop" />
25
26     <LinearLayout
27         android:layout_width="match_parent"
28         android:layout_height="wrap_content"
29         android:orientation="vertical"
30         android:padding="16dp">
31
32         <TextView
33             android:id="@+id/product_title"
34             android:layout_width="match_parent"
35             android:layout_height="wrap_content"

```

36	android:text="@string/shr_product_title"
37	android:textAlignment="center"
38	android:textAppearance="?attr/textAppearanceSubtitle2" />
39	
40	<TextView
41	android:id="@+id/product_price"
42	android:layout_width="match_parent"
43	android:layout_height="wrap_content"
44	android:text="@string/shr_product_description"
45	android:textAlignment="center"
46	android:textAppearance="?attr/textAppearanceBody2" />
47	</LinearLayout>
48	</LinearLayout>
49	</com.google.android.material.card.MaterialCardView>

Tabel 41. Source Code Soal 1 XML yang ketujuh

shr_staggered_product_card_third.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<com.google.android.material.card.MaterialCardView
	xmlns:android="http://schemas.android.com/apk/res/android"
3	xmlns:app="http://schemas.android.com/apk/res-auto"
4	android:layout_width="@dimen/shr_staggered_product_grid_card_width_portrait"
5	android:layout_height="match_parent"
6	app:cardBackgroundColor="@android:color/transparent"
7	app:cardElevation="0dp"
8	app:cardPreventCornerOverlap="true"
9	app:contentPaddingTop="@dimen/shr_staggered_product_grid_margin_top_third">
10	
11	<!-- Product card layout used in MDC-103 and onward -->
12	<LinearLayout

```

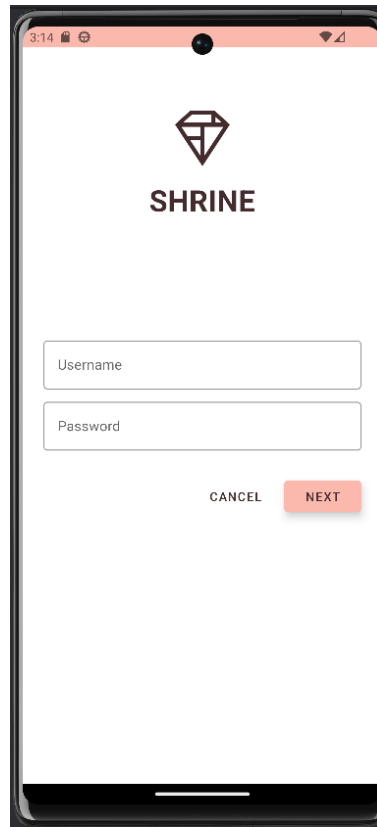
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:orientation="vertical">
16
17         <com.android.volley.toolbox.NetworkImageView
18             android:id="@+id/product_image"
19             android:layout_width="match_parent"
20             android:layout_height="@dimen/shr_staggered_product_
                grid_card_height_portrait"
21             android:background="?attr/colorPrimaryDark"
22             android:scaleType="centerCrop" />
23
24         <LinearLayout
25             android:layout_width="match_parent"
26             android:layout_height="wrap_content"
27             android:orientation="vertical"
28             android:padding="16dp">
29
30             <TextView
31                 android:id="@+id/product_title"
32                 android:layout_width="match_parent"
33                 android:layout_height="wrap_content"
34                 android:text="@string/shr_product_title"
35                 android:textAlignment="center"
36                 android:textAppearance="?attr/textAppearanceSubtitle
                    2" />
37
38                 <TextView
39                     android:id="@+id/product_price"
40                     android:layout_width="match_parent"
41                     android:layout_height="wrap_content"
42                     android:text="@string/shr_product_description"
43                     android:textAlignment="center"
44                     android:textAppearance="?attr/textAppearanceBody2"
45                     />
                </LinearLayout>

```

46	</LinearLayout>
47	</com.google.android.material.card.MaterialCardView>

Tabel 42. Source Code Soal 1 XML yang kedelean

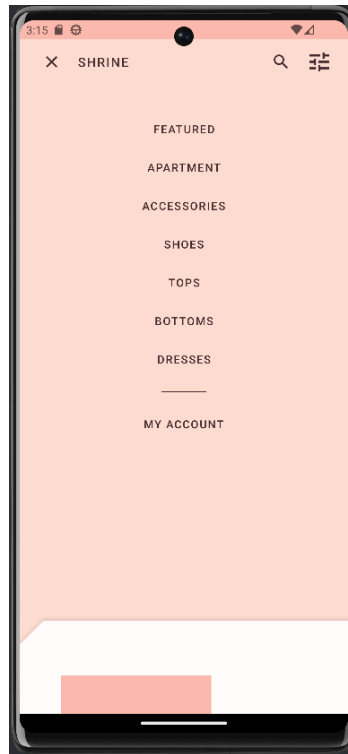
B. Output Program



Gambar 23. Screenshot Hasil Jawaban Soal 1 tampilan login



Gambar 24. Screenshot Hasil Jawaban Soal 1 tampilan di halaman utama



Gambar 25. Screenshot Hasil Jawaban Soal 1 tampilan menu

C. Pembahasan

Untuk file ShrineApplication.kt

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine.application`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class ShrineApplication: Ini adalah kelas yang mengimplementasikan `Application`, yang merupakan titik masuk utama untuk aplikasi Android. Ketika aplikasi Anda dimulai, metode `onCreate()` dari kelas ini akan dipanggil secara otomatis oleh sistem Android.

Companion Object: Terdapat objek companion yang menyediakan variabel instance. Variabel ini digunakan untuk menyimpan instansi dari `ShrineApplication` sehingga dapat diakses secara global di seluruh aplikasi. Penggunaan `private set` memastikan bahwa nilai dari instance hanya bisa diatur di dalam kelas `ShrineApplication`.

Metode `onCreate()`: Metode ini dipanggil saat aplikasi dimulai. Di dalamnya, `super.onCreate()` memanggil implementasi metode dari kelas induk (`Application`). Kemudian, `instance = this` menginisialisasi variabel `instance` dengan instansi saat ini dari `ShrineApplication`.

`AppCompatActivity`:

`BarisAppCompatActivity.setCompatVectorFromResourcesEnabled(true)`

mengaktifkan dukungan untuk menggunakan drawable vektor yang kompatibel pada aplikasi, sehingga memungkinkan penggunaan gambar vektor di semua versi Android.

Kesimpulan: Kelas `ShrineApplication` digunakan untuk menginisialisasi dan menyiapkan aplikasi Anda saat dimulai. Penggunaan companion object untuk `instance` memungkinkan Anda untuk dengan mudah mengakses konteks aplikasi dari mana saja dalam kode Anda.

Untuk file `ImageRequester.kt`

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine.network`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Object `ImageRequester`: Ini adalah objek Kotlin, yang merupakan singleton karena dideklarasikan dengan kata kunci `object`. Singleton memungkinkan satu instansi tunggal dari objek ini ada di seluruh aplikasi.

Properties:

`requestQueue`: Objek `RequestQueue` dari Volley yang digunakan untuk mengelola antrian permintaan jaringan.

`imageLoader`: Objek `ImageLoader` dari Volley yang digunakan untuk memuat gambar dari URL dan meng-cache gambar dalam memori.

`maxByteSize`: Ukuran maksimum cache dalam byte, dihitung berdasarkan ukuran layar perangkat untuk optimalisasi penggunaan memori.

Init Block: Blok `init` digunakan untuk inisialisasi properti saat objek `ImageRequester` pertama kali dibuat. Di dalamnya:

`context`: Mendapatkan konteks aplikasi menggunakan `ShrineApplication.instance`.

`requestQueue`: Membuat antrian permintaan baru menggunakan `Volley.newRequestQueue(context)` dan memulainya dengan `requestQueue.start()`.

maxByteSize: Menghitung ukuran maksimum cache dalam byte menggunakan calculateMaxByteSize(context).

imageLoader: Membuat ImageLoader yang menggunakan requestQueue dan mengimplementasikan cache gambar dengan LruCache.

Metode setImageFromUrl: Metode ini digunakan untuk menetapkan gambar dari URL ke NetworkImageView menggunakan imageLoader.

Metode calculateMaxByteSize: Metode ini menghitung ukuran maksimum cache dalam byte berdasarkan resolusi layar perangkat untuk memastikan efisiensi penggunaan memori.

Kesimpulan: Objek ImageRequester menyediakan fungsi untuk mengelola permintaan gambar secara efisien menggunakan Volley, termasuk pengaturan antrian permintaan dan caching gambar untuk meningkatkan kinerja aplikasi Android yang menggunakan jaringan.

Untuk file ProductEntry.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine.network). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class ProductEntry: Ini adalah kelas yang merepresentasikan entri produk dalam daftar produk. Ini memiliki beberapa properti:

title: Judul produk.

dynamicUrl: URL dinamis produk, diinisialisasi dari string dynamicUrl dan diubah menjadi objek Uri.

url: URL tetap produk.

price: Harga produk.

description: Deskripsi produk.

Constructor: Konstruktor primer (primary constructor) dari ProductEntry menginisialisasi properti title, dynamicUrl, url, price, dan description saat objek ProductEntry dibuat.

Companion Object:

`initProductEntryList(resources: Resources)`: Metode statis dalam companion object yang dimaksudkan untuk memuat data produk dari file JSON mentah yang terletak di `R.raw.products`.

`resources.openRawResource(R.raw.products)`: Membuka file JSON produk dari sumber daya mentah aplikasi.

`inputStream.bufferedReader().use(BufferedReader::readText)`: Membaca konten file JSON menjadi sebuah string menggunakan `BufferedReader`.

`Gson()`: Objek `Gson` untuk melakukan parsing JSON.

`TypeToken<ArrayList<ProductEntry>>() {}.type`: Membuat tipe data yang sesuai untuk daftar `ProductEntry`.

`gson.fromJson<List<ProductEntry>>(jsonProductsString, productListViewType)`: Mengkonversi JSON menjadi daftar objek `ProductEntry`.

Kesimpulan: Kelas `ProductEntry` digunakan untuk merepresentasikan entri produk dengan properti-properti yang sesuai. Metode dalam companion object-nya digunakan untuk memuat daftar produk dari file JSON yang tersedia dalam sumber daya aplikasi, menggunakan `Gson` untuk parsing data JSON ke dalam objek `ProductEntry`.

Untuk file `StaggeredProductCardRecyclerViewAdapter.kt`

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class `StaggeredProductCardRecyclerViewAdapter`: Kelas ini adalah adapter untuk `RecyclerView` yang menampilkan produk dalam grid asimetris. Kelas ini meng-extend `RecyclerView.Adapter` dan menggunakan `StaggeredProductCardViewHolder` sebagai `ViewHolder`-nya.

Constructor:

`private val productList: List<ProductEntry>?`: Konstruktor menerima daftar produk (`productList`) yang akan ditampilkan dalam `RecyclerView`. Daftar ini bisa null.

Methods:

`getItemViewType(position: Int)`: Mengembalikan tipe tampilan berdasarkan posisi item. Tipe tampilan dihitung sebagai `position % 3` untuk memberikan variasi tata letak asimetris.

`onCreateViewHolder(parent: ViewGroup, viewType: Int)`: Metode ini dipanggil saat RecyclerView membutuhkan ViewHolder baru. Berdasarkan viewType, metode ini menginflate tata letak yang sesuai (`shr_staggered_product_card_first`, `shr_staggered_product_card_second`, atau `shr_staggered_product_card_third`) dan mengembalikan `StaggeredProductCardViewHolder`.

`onBindViewHolder(holder: StaggeredProductCardViewHolder, position: Int)`: Metode ini mengikat data produk ke ViewHolder. Jika `productList` tidak null dan `position` valid, metode ini mengatur teks judul dan harga produk, serta memuat gambar produk menggunakan `ImageRequester`.

`getItemCount()`: Mengembalikan jumlah item dalam `productList`, atau 0 jika `productList` null.

`ViewHolder StaggeredProductCardViewHolder`: ViewHolder ini menyimpan referensi ke tampilan yang digunakan untuk menampilkan produk. Dalam metode `onBindViewHolder`, data produk diambil dari `productList` dan diatur pada ViewHolder ini.

Kesimpulan: `StaggeredProductCardRecyclerViewAdapter` adalah adapter yang dirancang untuk menampilkan produk dalam grid asimetris menggunakan RecyclerView. Adapter ini menyesuaikan tata letak item berdasarkan posisi mereka untuk memberikan tampilan yang menarik dan bervariasi. Metode `onCreateViewHolder` dan `onBindViewHolder` memastikan bahwa tampilan yang tepat digunakan untuk setiap item dan data produk diikat dengan benar pada tampilan tersebut.

Untuk File `StaggeredProductCardViewHolder.kt`

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class `StaggeredProductCardViewHolder`: Kelas ini adalah ViewHolder yang digunakan oleh `StaggeredProductCardRecyclerViewAdapter` untuk menampilkan item produk dalam RecyclerView. ViewHolder ini meng-extend `RecyclerView.ViewHolder` dan memiliki konstruktor yang mengambil satu parameter `itemView`, yang merupakan tampilan satu item produk dalam grid.

Properties:

`productImage`: Objek `NetworkImageView` yang digunakan untuk menampilkan gambar produk. Diinisialisasi dengan `itemView.findViewById(R.id.product_image)`,

yang mengambil referensi ke ImageView yang memiliki ID product_image dalam tata letak item.

productTitle: Objek TextView yang digunakan untuk menampilkan judul produk. Diinisialisasi dengan itemView.findViewById(R.id.product_title), yang mengambil referensi ke TextView yang memiliki ID product_title dalam tata letak item.

productPrice: Objek TextView yang digunakan untuk menampilkan harga produk. Diinisialisasi dengan itemView.findViewById(R.id.product_price), yang mengambil referensi ke TextView yang memiliki ID product_price dalam tata letak item.

Kesimpulan: StaggeredProductCardViewHolder adalah kelas yang berfungsi sebagai ViewHolder untuk menahan referensi ke elemen-elemen UI yang digunakan untuk menampilkan data produk dalam RecyclerView. Properti-propertinya (productImage, productTitle, dan productPrice) diinisialisasi dengan menggunakan metode findViewById pada itemView, yang merepresentasikan tampilan satu item produk dalam grid. ViewHolder ini digunakan oleh adapter (StaggeredProductCardRecyclerViewAdapter) untuk mengikat data produk ke tampilan yang sesuai saat item RecyclerView diperbarui atau ditampilkan.

Untuk File LoginFragment.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class LoginFragment: Kelas ini merupakan subclass dari Fragment yang digunakan untuk menampilkan layar login Shrine. Fragment ini mengatur tata letak dan interaksi yang terjadi di dalamnya.

Metode onCreateView:

Metode ini dipanggil ketika fragment pertama kali membuat tampilannya (View).

inflater.inflate(R.layout.shr_login_fragment, container, false): Memuat tata letak fragment dari file XML dengan nama shr_login_fragment.

view.findViewById<Button>(R.id.next_button).setOnClickListener { ... }: Menetapkan listener untuk tombol "Next". Jika panjang kata sandi kurang dari 8 karakter, kesalahan ditetapkan pada TextInputLayout. Jika tidak, kesalahan dibersihkan dan navigasi dilakukan ke ProductGridFragment.

view.findViewById<EditText>(R.id.password_edit_text).setOnKeyListener { _, _, _ -> ... }: Menetapkan listener untuk EditText kata sandi. Menghapus kesalahan jika lebih dari 8 karakter diketik.

Metode `isPasswordValid`:

Metode ini memeriksa apakah teks yang diberikan (kata sandi) memiliki panjang setidaknya 8 karakter.

Kesimpulan: `LoginFragment` adalah bagian dari antarmuka pengguna `Shrine` yang bertanggung jawab untuk menampilkan layar login, memvalidasi kata sandi, dan menangani navigasi antar fragmen. Ini menunjukkan praktik penggunaan fragment, penggunaan XML untuk tata letak, penggunaan listener untuk interaksi pengguna, dan penggunaan `Editable` untuk memeriksa validitas kata sandi.

Untuk File `MainActivity.kt`

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class `MainActivity`: Kelas ini merupakan subclass dari `AppCompatActivity` dan mengimplementasikan antarmuka `NavigationHost`.

`onCreate` Method:

`super.onCreate(savedInstanceState)`: Memanggil metode `onCreate` dari kelas induk `AppCompatActivity`.

`setContentView(R.layout.shr_main_activity)`: Mengatur tampilan konten dari aktivitas ini menggunakan layout XML `shr_main_activity`.

`if (savedInstanceState == null) { ... }`: Memeriksa apakah `savedInstanceState` adalah null. Jika ya, ini berarti aktivitas baru dibuat, dan kita menambahkan `LoginFragment` ke `FragmentManager` menggunakan transaksi fragmen.

`navigateTo` Method:

`override fun navigateTo(fragment: Fragment, addToBackStack: Boolean)`: Metode ini mengimplementasikan metode dari antarmuka `NavigationHost`.

`val transaction = supportFragmentManager.beginTransaction().replace(R.id.container, fragment)`: Membuat transaksi fragmen baru yang menggantikan konten di dalam `R.id.container` dengan fragmen baru.

`if (addToBackStack) { transaction.addToBackStack(null) }`: Jika `addToBackStack` bernilai true, fragmen saat ini ditambahkan ke backstack, memungkinkan pengguna untuk kembali ke fragmen sebelumnya dengan menekan tombol "back".

`transaction.commit()`: Melakukan transaksi fragmen.

Kesimpulan: MainActivity adalah aktivitas utama yang mengelola dan menavigasi antara berbagai fragmen dalam aplikasi. Ini mengatur tampilan utama menggunakan layout XML dan menambahkan LoginFragment sebagai fragmen awal jika aktivitas baru dibuat. Metode navigateTo memungkinkan navigasi ke fragmen lain, dengan opsi untuk menambahkannya ke backstack untuk navigasi mundur. Dengan mengimplementasikan antarmuka NavigationHost, MainActivity memfasilitasi navigasi yang fleksibel dalam aplikasi.

Untuk file NavigationHost.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Interface NavigationHost:

Ini adalah sebuah antarmuka Kotlin yang mendefinisikan kontrak untuk host (biasanya sebuah Activity) yang mampu menampilkan fragmen dan merespons peristiwa navigasi.

Antarmuka ini memiliki satu metode:

fun navigateTo(fragment: Fragment, addToBackstack: Boolean): Metode ini digunakan untuk memicu navigasi ke fragmen yang ditentukan. Parameter fragment adalah fragmen yang akan ditampilkan, sedangkan addToBackstack menentukan apakah transaksi navigasi harus ditambahkan ke backstack untuk navigasi mundur.

Kesimpulan: Antarmuka NavigationHost memberikan abstraksi yang diperlukan untuk mengelola navigasi antara fragmen dalam aplikasi Android. Dengan menggunakan antarmuka ini, sebuah aktivitas atau komponen lain dapat mengimplementasikan metode navigateTo untuk menangani perpindahan antar fragmen dengan cara yang terstruktur dan dapat dibalikkan (reversible). Ini meningkatkan modularitas dan pemisahan tugas dalam pengembangan aplikasi, memungkinkan komponen-komponen aplikasi untuk berkomunikasi dengan cara yang terdefinisi dengan baik.

Untuk file NavigationIconClickListener.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class NavigationIconClickListener:

Kelas ini mengimplementasikan View.OnClickListener dan dibuat dengan anotasi @JvmOverloads untuk kompatibilitas Java.

Konstruktor menerima beberapa parameter:

context: Konteks aplikasi atau aktivitas yang digunakan untuk mengakses sumber daya dan layanan Android.

sheet: View yang akan dianimasikan (lembaran produk grid).

interpolator: Objek interpolasi untuk mengatur animasi.

openIcon dan closeIcon: Drawable untuk ikon terbuka dan tertutup, yang digunakan untuk memperbarui ikon di toolbar.

Properties and Initialization:

animatorSet: Objek AnimatorSet untuk mengelola animasi.

height: Tinggi layar perangkat, diambil dari DisplayMetrics untuk menghitung pergeseran animasi.

onClick Method:

Metode ini dipanggil saat view yang di-attach OnClickListener ini ditekan.

backdropShown disetel berdasarkan keadaan sebelumnya (true/false).

Animasi yang ada dibatalkan dan dihapus, dan kemudian diperbarui dengan animasi baru berdasarkan keadaan backdropShown.

ObjectAnimator digunakan untuk menganimasikan sheet bergerak naik atau turun sesuai keadaan backdropShown.

updateIcon Method:

Metode ini memperbarui ikon navigasi di toolbar (ImageView) berdasarkan keadaan backdropShown.

Jika backdropShown adalah true, ikon ditetapkan sebagai closeIcon, dan sebaliknya untuk openIcon.

Kesimpulan: NavigationIconClickListener adalah implementasi khusus dari OnClickListener yang digunakan untuk mengendalikan animasi dan perubahan ikon saat ikon navigasi di toolbar ditekan. Ini menangani pengaturan dan animasi dari sheet (lembaran) produk grid, serta memungkinkan penyesuaian interpolasi animasi dan pengaturan ikon yang dinamis berdasarkan keadaan aplikasi.

Untuk file ProductCardRecyclerViewAdapter.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class ProductCardRecyclerViewAdapter:

Kelas ini meng-extend RecyclerView.Adapter dengan ProductCardViewHolder sebagai ViewHolder-nya.

Konstruktur internal (internal constructor) menerima parameter productList, yang merupakan daftar objek ProductEntry yang akan ditampilkan dalam RecyclerView.

Methods:

onCreateViewHolder(parent: ViewGroup, viewType: Int): Metode ini dipanggil saat RecyclerView membutuhkan ViewHolder baru untuk item. Dalam metode ini:

LayoutInflater.from(parent.context).inflate(R.layout.shr_product_card, parent, false): Menginflate tata letak item produk (shr_product_card) dari file XML ke dalam ViewGroup parent.

Mengembalikan ProductCardViewHolder yang diinisialisasi dengan layoutView.

onBindViewHolder(holder: ProductCardViewHolder, position: Int): Metode ini mengikat data produk ke ViewHolder untuk item di posisi tertentu. Dalam metode ini:

Memeriksa apakah position valid dalam productList.

Mengambil produk dari productList berdasarkan position.

Mengatur teks judul dan harga produk pada ViewHolder.

Menggunakan ImageRequester untuk memuat gambar produk dari URL dan menampilkannya pada holder.productImage.

getItemCount(): Metode ini mengembalikan jumlah item dalam productList.

Kesimpulan: ProductCardRecyclerViewAdapter adalah adapter yang digunakan untuk menghubungkan data produk (ProductEntry) dengan tampilan yang ditampilkan dalam RecyclerView. Ini menyediakan implementasi untuk mengatur ViewHolder baru, mengikat data produk ke ViewHolder, dan menghitung jumlah total item yang akan ditampilkan. Dengan menggunakan adapter ini, aplikasi dapat menampilkan daftar produk dalam grid sederhana dengan efisien dan dinamis di layar perangkat Android.

Untuk file ProductCardViewHolder.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class ProductCardViewHolder:

Kelas ini meng-extend RecyclerView.ViewHolder dan menerima satu parameter itemView yang merupakan tampilan untuk satu item dalam RecyclerView.

Dalam konstruktor (constructor), itemView di-pass ke superclass RecyclerView.ViewHolder menggunakan super(itemView) untuk menginisialisasi ViewHolder dengan tampilan yang diberikan.

Member Variables:

productImage: NetworkImageView yang merupakan komponen untuk menampilkan gambar produk. Inisialisasinya menggunakan findViewById untuk menemukan view dengan ID product_image di dalam itemView.

productTitle: TextView yang merupakan komponen untuk menampilkan judul produk. Inisialisasinya menggunakan findViewById untuk menemukan view dengan ID product_title di dalam itemView.

productPrice: TextView yang merupakan komponen untuk menampilkan harga produk. Inisialisasinya menggunakan findViewById untuk menemukan view dengan ID product_price di dalam itemView.

Kesimpulan: ProductCardViewHolder bertanggung jawab untuk menginisialisasi dan menyimpan referensi ke komponen tampilan dalam satu item produk dalam RecyclerView. Dengan memiliki referensi ke komponen-komponen ini, ViewHolder memungkinkan adapter untuk mengatur data produk ke tampilan yang sesuai saat menangani pengelolaan daftar produk yang ditampilkan dalam RecyclerView. Ini adalah pendekatan yang umum digunakan dalam pengembangan aplikasi Android untuk meningkatkan kinerja dan pengelolaan tampilan yang kompleks.

Untuk file ProductGridFragment.kt

Package Declaration: Kode dimulai dengan deklarasi paket (package com.example.praktikum.frly.kotlin.shrine). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class ProductGridFragment:

Kelas ini meng-extend Fragment dan bertanggung jawab untuk menampilkan tampilan grid produk dalam aplikasi Shrine.

Pada `onCreate()`, dipanggil `setHasOptionsMenu(true)` untuk memberitahukan bahwa Fragment memiliki menu opsional.

Lifecycle Methods:

`onCreateView()`: Method ini meng-inflate layout untuk Fragment menggunakan `R.layout.shr_product_grid_fragment`.

Toolbar dan RecyclerView diinisialisasi dari layout yang di-inflate.

Toolbar di-set sebagai ActionBar menggunakan `setSupportActionBar(app_bar)` dan ditambahkan `NavigationOnClickListener` untuk menangani interaksi klik pada ikon navigasi.

RecyclerView diatur dengan `GridLayoutManager` yang memiliki 2 kolom horizontal. `StaggeredProductCardRecyclerViewAdapter` digunakan sebagai adapter untuk menampilkan daftar produk.

`ProductGridItemDecoration` ditambahkan sebagai dekorasi untuk menentukan spasi antar item dalam RecyclerView.

Background dari `NestedScrollView` (`product_grid`) di-set menggunakan drawable dari resources.

`onCreateOptionsMenu()`: Method ini digunakan untuk inflate menu toolbar dari `R.menu.shr_toolbar_menu` yang kemudian ditampilkan di Toolbar Fragment.

Kesimpulan: `ProductGridFragment` adalah bagian dari aplikasi Shrine yang menangani tampilan grid produk menggunakan RecyclerView. Fragment ini mengatur UI elemen seperti Toolbar, RecyclerView, dan menampilkan daftar produk dengan menggunakan adapter khusus. Dengan cara ini, Fragment memungkinkan pengguna untuk menavigasi produk secara interaktif dan memanfaatkan fitur toolbar untuk akses menu tambahan.

Untuk file `ProductGridItemDecoration.kt`

Package Declaration: Kode dimulai dengan deklarasi paket (`package com.example.praktikum.frly.kotlin.shrine`). Ini menunjukkan lokasi dari file kelas di dalam proyek.

Class `ProductGridItemDecoration`:

Kelas ini meng-extend `RecyclerView.ItemDecoration`, yang digunakan untuk menambahkan dekorasi visual ke item-item dalam RecyclerView.

Konstruktor menerima dua parameter: `largePadding` dan `smallPadding`, yang digunakan untuk menentukan jumlah padding yang diberikan ke item-item dalam `RecyclerView`.

Method `getItemOffsets()`:

Override dari method ini digunakan untuk menentukan offset (padding) yang diberikan ke item-item dalam `RecyclerView`.

`outRect.left` di-set dengan nilai `smallPadding`, yang menambahkan sedikit padding di sebelah kiri setiap item.

`outRect.right` di-set dengan nilai `largePadding`, yang menambahkan banyak padding di sebelah kanan setiap item.

Tujuan:

Tujuan dari `ProductGridItemDecoration` adalah untuk memperindah tampilan grid produk dengan menambahkan padding khusus di sekitar setiap item dalam `RecyclerView`.

Padding ini membantu memisahkan visual antara item-item, memberikan ruang yang cukup agar item-item tersebut terlihat terpisah dan mudah diidentifikasi.

Kesimpulan: `ProductGridItemDecoration` adalah komponen penting dalam desain visual `ProductGridFragment`, yang meningkatkan estetika tampilan grid produk dengan menambahkan padding yang sesuai di sekitar setiap item dalam `RecyclerView`. Dengan cara ini, penggunaan `ItemDecoration` memungkinkan pengembang untuk secara fleksibel mengontrol ruang dan jarak antara item-item dalam `RecyclerView`.

Untuk file `shr_backdrop.xml`

Deklarasi XML: Baris pertama `<?xml version="1.0" encoding="utf-8"?>` adalah deklarasi XML standar yang menyatakan versi XML dan encoding yang digunakan.

Elemen `merge`: Elemen `<merge>` digunakan untuk menggabungkan tampilan dari beberapa layout yang berbeda ke dalam satu tampilan. Ini biasanya digunakan dalam situasi di mana Anda ingin menyertakan beberapa layout ke dalam satu layout induk tanpa wrapping tambahan.

Komponen UI:

Tujuh elemen `<com.google.android.material.button.MaterialButton>` digunakan untuk menampilkan tombol dengan gaya yang didefinisikan dalam `style="@style/Widget.Shrine.Button.TextButton"`. Setiap tombol memiliki lebar dan tinggi yang disesuaikan dengan teks yang ditetapkan menggunakan atribut `android:layout_width="wrap_content"` dan `android:layout_height="wrap_content"`.

Satu elemen `<View>` digunakan untuk menampilkan garis horizontal dengan lebar 56dp, tinggi 1dp, dan margin 16dp. Garis ini memiliki latar belakang yang menggunakan warna primer teks sistem, didefinisikan dengan atribut `android:background="?android:attr/textColorPrimary"`.

Atribut Tambahan:

Setiap elemen `MaterialButton` menggunakan atribut `android:text` untuk menetapkan teks tombol berdasarkan nilai dari string resources seperti `@string/shr_featured_label`.

Elemen `<View>` menggunakan atribut-atribut seperti `android:layout_width`, `android:layout_height`, `android:layout_margin`, dan `android:background` untuk menentukan dimensi, margin, dan latar belakang dari garis horizontal.

Gaya (style): Tombol-tombol `MaterialButton` menggunakan gaya yang didefinisikan dalam `@style/Widget.Shrine.Button.TextButton`. Gaya ini bisa didefinisikan di file `styles.xml` dalam folder `res/values`.

Tujuan: Layout ini digunakan untuk menampilkan beberapa tombol `MaterialButton` yang mewakili kategori produk atau menu navigasi dalam aplikasi. Garis horizontal digunakan untuk memisahkan antara kategori produk dan bagian akun dalam antarmuka pengguna.

Kesimpulan: File XML ini merupakan bagian dari antarmuka pengguna dalam aplikasi Android yang menggunakan Material Design komponen untuk menampilkan tombol-tombol kategori produk dan bagian akun, dengan tambahan garis horizontal sebagai elemen visual pemisah. Penyertaan semua elemen dalam satu file dengan `<merge>` memungkinkan untuk memadukan tampilan dari beberapa layout yang berbeda secara efisien dalam satu tampilan.

Untuk File `shr_login_fragment.xml`

Deklarasi XML: Baris pertama `<?xml version="1.0" encoding="utf-8"?>` adalah deklarasi XML standar yang menyatakan versi XML dan encoding yang digunakan.

ScrollView: Elemen <ScrollView> digunakan sebagai kontainer utama yang memungkinkan pengguna untuk menggulir konten jika konten melebihi area tampilan yang tersedia. Atribut yang digunakan di antaranya:

android:layout_width dan android:layout_height diatur ke match_parent, sehingga ScrollView mengisi seluruh ruang yang tersedia dalam parent-nya.

android:background menetapkan warna latar belakang menggunakan nilai dari resource warna (@color/loginPageBackgroundColor).

LinearLayout: Elemen <LinearLayout> adalah child dari ScrollView dan digunakan untuk menyusun elemen-elemen secara vertikal. Atribut-atribut yang digunakan di antaranya:

android:layout_width dan android:layout_height diatur ke match_parent dan wrap_content berturut-turut.

android:clipChildren dan android:clipToPadding diatur ke false untuk memastikan bahwa elemen-elemen di dalamnya tidak terpotong oleh batas ScrollView.

android:orientation diatur ke vertical untuk menata elemen-elemen secara vertikal.

android:padding dan android:paddingTop menetapkan jarak padding dari sisi luar LinearLayout.

ImageView: Elemen <ImageView> menampilkan gambar logo dengan beberapa atribut seperti:

android:layout_width dan android:layout_height untuk menetapkan ukuran gambar.

android:layout_gravity digunakan untuk menengahkan gambar secara horizontal di dalam parent-nya.

app:srcCompat menetapkan gambar menggunakan resource drawable (@drawable/shr_logo).

android:contentDescription memberikan deskripsi konten untuk aksesibilitas (@string/shr_logo_content_description).

app:tint memberikan efek pewarnaan untuk gambar menggunakan nilai warna dari atribut sistem (?android:attr/textColorPrimary).

TextView: Elemen <TextView> menampilkan teks dengan atribut-atribut seperti:

android:layout_width dan android:layout_height untuk menetapkan ukuran teks.

`android:layout_gravity` untuk menengahkan teks secara horizontal di dalam parent-nya.

`android:text` menetapkan teks menggunakan resource string (`@string/shr_app_name`).

`android:textAppearance` menetapkan penampilan teks menggunakan gaya yang didefinisikan (`@style/TextAppearance.Shrine.Title`).

TextInputLayout dan TextInputEditText: Elemen `<TextInputLayout>` dan `<TextInputEditText>` digunakan untuk menampilkan dan mengelola input teks seperti username dan password. `TextInputLayout` menyediakan label floating dan ruang input yang terintegrasi dengan baik. Atribut-atribut seperti `android:hint` untuk label placeholder dan `android:inputType` untuk jenis input ditetapkan di dalam `TextInputEditText`.

RelativeLayout: Elemen `<RelativeLayout>` digunakan untuk menempatkan dua tombol secara relatif di bagian bawah layout. Atribut seperti `android:layout_width`, `android:layout_height`, dan `android:layout_alignParentEnd` digunakan untuk menyesuaikan penempatan tombol.

MaterialButton: Elemen `<com.google.android.material.button.MaterialButton>` digunakan untuk menampilkan tombol Material Design dengan gaya yang telah ditetapkan (`@style/Widget.Shrine.Button` dan `@style/Widget.Shrine.Button.TextButton`). Atribut `android:text` menetapkan teks tombol menggunakan nilai dari resource string.

Interaksi dan Fungsionalitas: Tombol-tombol seperti "Next" dan "Cancel" digunakan untuk interaksi pengguna, yang diimplementasikan dalam kelas `LoginFragment`.

Kesimpulan: File XML ini mendefinisikan tata letak UI untuk fragment login dalam aplikasi Android menggunakan `ScrollView` sebagai wadah utama untuk konten scrollable. Elemen-elemen seperti `ImageView`, `TextView`, `TextInputLayout`, dan `MaterialButton` digunakan untuk menampilkan dan mengelola input serta interaksi pengguna dengan antarmuka login.

Untuk File `fragment_overview.xml`

Root Element: `<layout>` adalah elemen root (akar) dari file XML ini, menandakan bahwa layout ini menggunakan Data Binding.

Namespaces:

`xmlns:android="http://schemas.android.com/apk/res/android"`: Namespace Android untuk atribut dan elemen Android.

`xmlns:app="http://schemas.android.com/apk/res-auto"`: Namespace AppCompatActivity untuk atribut tambahan yang terkait dengan kompatibilitas aplikasi.

`xmlns:tools="http://schemas.android.com/tools"`: Namespace untuk alat bantu pengembangan Android Studio (tools).

Data Binding:

`<data>`: Bagian ini mendefinisikan variabel yang digunakan dalam Data Binding.

`<variable>`: Mendefinisikan variabel `viewModel` dengan tipe `OverviewViewModel`. Variabel ini akan diikat (bind) dengan `ViewModel` yang terkait dari kelas `OverviewViewModel`.

Constraint Layout:

`<ConstraintLayout>`: Digunakan sebagai root container, mengatur tampilan dengan konstrain untuk memastikan tata letak yang responsif.

Views:

`<RecyclerView>`: Menampilkan grid foto-foto properti Mars menggunakan `app:layoutManager` yang diikat ke `viewModel.properties`. Ini memanfaatkan Data Binding untuk menampilkan daftar properti secara dinamis.

`app:layoutManager`: Mengatur `GridLayoutManager` untuk mengatur tata letak grid.

`app:spanCount`: Menentukan jumlah kolom dalam grid (dalam hal ini, 2 kolom).

`tools:itemCount` dan `tools:listitem`: Digunakan oleh Android Studio untuk desain UI di mode desain.

`<ImageView>`: Menampilkan status dari API Mars menggunakan `app:marsApiStatus` yang diikat ke `viewModel.status`. Ini memanfaatkan Data Binding untuk menampilkan status API secara dinamis.

Padding and ClipToPadding:

`android:padding="6dp"`: Memberikan padding sebesar 6dp pada `RecyclerView`.

`android:clipToPadding="false"`: Mengatur agar padding tidak memengaruhi efek scroll di dalam `RecyclerView`.

Tools Namespace:

Digunakan untuk memberikan nilai dummy pada tampilan saat merancang UI di Android Studio (`tools:itemCount`, `tools:listitem`).

File XML ini digunakan dalam konteks MainActivity atau mungkin juga OverviewFragment, yang terkait dengan menampilkan daftar properti Mars dan status API menggunakan Data Binding untuk mempermudah pengelolaan dan interaksi antara tampilan dan data aplikasi.

Untuk File shr_main_activity.xml

Deklarasi XML: Baris pertama `<?xml version="1.0" encoding="utf-8"?>` adalah deklarasi XML standar yang menyatakan versi XML dan encoding yang digunakan.

FrameLayout: Elemen `<FrameLayout>` digunakan sebagai kontainer utama untuk menempatkan fragment atau layout lainnya di dalamnya. Atribut-atribut yang digunakan di antaranya:

`xmlns:android="http://schemas.android.com/apk/res/android"` dan `xmlns:tools="http://schemas.android.com/tools"` adalah namespace yang diperlukan untuk penggunaan atribut dan elemen dalam XML Android.

`android:id="@+id/container"` menetapkan ID unik untuk FrameLayout ini, yang nantinya digunakan untuk menempatkan fragment atau layout lainnya secara dinamis dari dalam kode.

`android:layout_width="match_parent"` dan `android:layout_height="match_parent"` mengatur ukuran FrameLayout agar mengisi seluruh area yang tersedia dalam parent-nya, baik dalam hal lebar maupun tinggi.

`tools:context="com.example.praktikum.frly.kotlin.shrine.MainActivity"` digunakan oleh Android Studio untuk tujuan desain UI dan memeriksa referensi context.

Fungsionalitas: FrameLayout ini tidak menampilkan elemen visual apa pun secara langsung dalam XML-nya. Sebagai gantinya, dalam praktiknya, aktivitas MainActivity akan menempatkan fragment pertama kali di dalam FrameLayout ini saat dibuat (seperti yang dilakukan dalam metode `onCreate` dalam kode Kotlin yang telah diberikan sebelumnya).

Kesimpulan: File XML ini menentukan layout dasar untuk aktivitas MainActivity, dengan menggunakan FrameLayout sebagai kontainer utama yang akan menampung fragment-fragment aplikasi. Penggunaan FrameLayout ini memungkinkan penggantian fragment-fragment tanpa harus mengubah layout utama atau menginisialisasi ulang seluruh layout aktivitas.

Untuk File `shr_product_card.xml`

Deklarasi XML: File dimulai dengan deklarasi XML standar yang menunjukkan versi XML dan encoding yang digunakan.

MaterialCardView: Ini adalah elemen utama dalam file XML, yang mewakili kartu material. Beberapa atribut yang digunakan di sini adalah:

`xmlns:android="http://schemas.android.com/apk/res/android"` dan `xmlns:app="http://schemas.android.com/apk/res-auto"` adalah namespace yang diperlukan untuk menggunakan atribut dan komponen Android dan Material Design.

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi kartu sehingga lebarnya mengisi seluruh area yang tersedia, sedangkan tingginya menyesuaikan dengan konten di dalamnya.

`app:cardBackgroundColor="@android:color/transparent"` menetapkan warna latar belakang kartu menjadi transparan.

`app:cardElevation="0dp"` menghilangkan bayangan dari kartu dengan mengatur elevasi menjadi nol.

`app:cardPreventCornerOverlap="true"` memastikan sudut kartu tidak tumpang tindih.

LinearLayout: Ini adalah kontainer untuk tata letak dalam kartu, dengan orientasi vertikal. Beberapa elemen di dalamnya meliputi:

NetworkImageView: Digunakan untuk menampilkan gambar produk. Atribut `android:id="@+id/product_image"` digunakan untuk mengidentifikasi elemen dalam kode Kotlin yang terkait.

LinearLayout: Kontainer lain untuk judul dan harga produk dengan orientasi vertikal.

TextView `product_title`: Menampilkan judul produk.

TextView `product_price`: Menampilkan harga produk.

Penggunaan: Kartu ini dirancang untuk menampilkan informasi produk secara terstruktur dan estetik, sesuai dengan pedoman desain Material Design. Penggunaan MaterialCardView memungkinkan kartu untuk memiliki penampilan yang konsisten dengan aplikasi Android modern yang mengikuti pedoman desain terbaru.

Tujuan: Kartu seperti ini sangat berguna dalam aplikasi e-commerce atau aplikasi yang menampilkan daftar produk, karena memberikan tampilan yang bersih, mudah dipahami, dan estetik bagi pengguna.

Untuk shr_product_grid_fragment.xml

Deklarasi XML: File dimulai dengan deklarasi XML standar yang menunjukkan versi XML dan encoding yang digunakan.

FrameLayout: Ini adalah elemen utama dalam file XML yang digunakan sebagai kontainer untuk menempatkan beberapa tampilan secara berlapis. Beberapa atribut yang digunakan di sini adalah:

xmlns:android="http://schemas.android.com/apk/res/android",
xmlns:app="http://schemas.android.com/apk/res-auto",
xmlns:tools="http://schemas.android.com/tools" adalah namespace yang diperlukan untuk menggunakan atribut dan komponen Android.

android:layout_width="match_parent" dan android:layout_height="match_parent" mengatur dimensi FrameLayout agar mengisi seluruh area layar.

android:clipChildren="false" dan android:clipToPadding="false" menonaktifkan klipping terhadap elemen-elemen yang berada di dalam FrameLayout, memungkinkan elemen-elemen tersebut untuk melewati batas FrameLayout.

tools:context="com.example.praktikum.frly.kotlin.shrine.ProductGridFragment" menyediakan konteks untuk alat pengembangan Android Studio.

LinearLayout (Backdrop): Ini adalah kontainer untuk latar belakang fragment dengan gaya yang ditentukan oleh Widget.Shrine.Backdrop. Beberapa atribut yang digunakan di sini adalah:

style="@style/Widget.Shrine.Backdrop" menetapkan gaya yang telah didefinisikan sebelumnya untuk mengatur tampilan latar belakang.

android:gravity="center_horizontal" mengatur posisi horizontal isi dari LinearLayout menjadi di tengah.

android:orientation="vertical" mengatur orientasi kontainer menjadi vertikal.

android:paddingTop="88dp" memberikan padding di bagian atas LinearLayout.

Include Layout: Memasukkan layout shr_backdrop ke dalam LinearLayout. Ini memungkinkan untuk memanfaatkan kembali layout yang sudah ada tanpa harus menulis ulang.

AppBarLayout dan Toolbar: AppBarLayout bersama dengan Toolbar menyediakan bilah aplikasi di bagian atas layar. Beberapa atribut yang digunakan di sini adalah:

`app:elevation="0dp"` menghilangkan bayangan dari `AppBarLayout` dengan mengatur elevasi menjadi nol.

`android:paddingStart`, `android:paddingLeft`, `android:paddingEnd`, `android:paddingRight` menetapkan padding di sisi kiri dan kanan toolbar.

`app:contentInsetStart="0dp"` mengatur inset konten di sisi start toolbar menjadi nol.

`app:navigationIcon="@drawable/shr_branded_menu"` menetapkan ikon navigasi untuk toolbar.

`app:title="@string/shr_app_name"` menetapkan judul aplikasi untuk toolbar.

NestedScrollView: Ini adalah komponen yang mengelilingi `RecyclerView` untuk mendukung perilaku gulir yang diperlukan ketika konten di dalamnya lebih panjang dari ukuran layar. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_grid"` memberikan ID kepada `NestedScrollView` untuk referensi di kode Kotlin.

`android:layout_marginTop="56dp"` memberikan margin atas untuk memisahkan konten dari `AppBarLayout`.

`android:background="@color/productGridBackgroundColor"` menetapkan warna latar belakang untuk `NestedScrollView`.

`android:elevation="8dp"` memberikan elevasi untuk menampilkan bayangan pada `NestedScrollView`.

`app:layout_behavior="@string/appbar_scrolling_view_behavior"` mengaitkan perilaku gulir `NestedScrollView` dengan `AppBarLayout` sehingga `AppBar` tetap terlihat saat konten digulir.

RecyclerView: Ini adalah komponen untuk menampilkan daftar produk dalam grid. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/recycler_view"` memberikan ID kepada `RecyclerView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="match_parent"` mengatur dimensi `RecyclerView` agar mengisi seluruh area yang tersedia dalam `NestedScrollView`.

File XML ini dirancang untuk menampilkan tampilan grid produk dalam aplikasi Android dengan menggunakan Material Design components seperti `AppBarLayout`,

Toolbar, dan RecyclerView, serta mengoptimalkan tata letak dengan menggunakan FrameLayout dan NestedScrollView.

Untuk shr_staggered_product_card_first.xml

Deklarasi XML: File dimulai dengan deklarasi XML standar yang menunjukkan versi XML dan encoding yang digunakan.

MaterialCardView: Ini adalah elemen utama dalam file XML yang digunakan untuk menampilkan kartu produk dengan tampilan yang konsisten dan elegan. Beberapa atribut yang digunakan di sini adalah:

`xmlns:android="http://schemas.android.com/apk/res/android"` dan `xmlns:app="http://schemas.android.com/apk/res-auto"` adalah namespace yang diperlukan untuk menggunakan atribut dan komponen Android serta Material Design.

`android:layout_width="wrap_content"` dan `android:layout_height="match_parent"` mengatur dimensi MaterialCardView agar lebar sesuai dengan konten (`wrap_content`) dan tinggi mengisi seluruh ketinggian parent.

`app:cardBackgroundColor="@android:color/transparent"` menetapkan warna latar belakang kartu menjadi transparan.

`app:cardElevation="0dp"` menghilangkan elevasi bayangan dari kartu dengan mengatur nilai elevasi menjadi nol.

`app:cardPreventCornerOverlap="true"` mengatur agar sudut kartu tidak tumpang tindih satu sama lain.

LinearLayout: Ini adalah kontainer untuk mengatur tampilan dalam kartu produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="@dimen/shr_staggered_product_grid_card_width_landscape"` mengatur lebar LinearLayout sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:layout_height="wrap_content"` mengatur tinggi LinearLayout agar mengikuti konten yang ada di dalamnya.

`android:layout_marginStart="@dimen/shr_staggered_product_grid_spacing_small"` dan

`android:layout_marginLeft="@dimen/shr_staggered_product_grid_spacing_small"` memberikan margin di sisi kiri LinearLayout sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

NetworkImageView: Ini adalah komponen untuk menampilkan gambar produk dari jaringan. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_image"` memberikan ID kepada `ImageView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="@dimen/shr_staggered_product_grid_card_height_landscape"` mengatur dimensi `ImageView` agar lebar mengisi parent dan tinggi sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:background="?attr/colorPrimaryDark"` menetapkan warna latar belakang `ImageView` menggunakan atribut warna dari tema aplikasi.

`android:scaleType="centerCrop"` mengatur jenis skala gambar agar selalu terpotong jika diperlukan untuk mengisi `ImageView` secara penuh.

LinearLayout (Informasi Produk): Ini adalah kontainer untuk informasi detail produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `LinearLayout` agar lebar mengisi parent dan tinggi mengikuti konten di dalamnya.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

`android:padding="16dp"` memberikan padding di dalam `LinearLayout` agar konten terlihat lebih teratur.

TextView (Judul Produk dan Harga): Ini adalah komponen untuk menampilkan judul dan harga produk. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_title"` dan `android:id="@+id/product_price"` memberikan ID kepada `TextView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `TextView` agar lebar mengisi parent dan tinggi mengikuti konten teks.

`android:text="@string/shr_product_title"` dan
`android:text="@string/shr_product_description"` mengatur teks yang ditampilkan menggunakan string yang telah didefinisikan sebelumnya di dalam `strings.xml`.

`android:textAlignment="center"` mengatur teks agar berada di tengah secara horizontal.

File XML ini dirancang untuk menampilkan kartu produk dalam grid dengan menggunakan `MaterialCardView` dan mengatur tata letaknya menggunakan `LinearLayout`. Ini adalah pendekatan yang umum digunakan dalam pengembangan aplikasi Android untuk menampilkan item dalam daftar atau grid dengan konsisten dan efisien.

Untuk `shr_staggered_product_card_second.xml`

Deklarasi XML: File dimulai dengan deklarasi XML standar yang menunjukkan versi XML dan encoding yang digunakan.

`MaterialCardView`: Ini adalah elemen utama dalam file XML yang digunakan untuk menampilkan kartu produk dengan tampilan yang konsisten dan elegan. Beberapa atribut yang digunakan di sini adalah:

`xmlns:android="http://schemas.android.com/apk/res/android"` dan
`xmlns:app="http://schemas.android.com/apk/res-auto"` adalah namespace yang diperlukan untuk menggunakan atribut dan komponen Android serta Material Design.

`android:layout_width="wrap_content"` dan `android:layout_height="match_parent"` mengatur dimensi `MaterialCardView` agar lebar sesuai dengan konten (`wrap_content`) dan tinggi mengisi seluruh ketinggian parent.

`app:cardBackgroundColor="@android:color/transparent"` menetapkan warna latar belakang kartu menjadi transparan.

`app:cardElevation="0dp"` menghilangkan elevasi bayangan dari kartu dengan mengatur nilai elevasi menjadi nol.

`app:cardPreventCornerOverlap="true"` mengatur agar sudut kartu tidak tumpang tindih satu sama lain.

`app:contentPaddingTop="@dimen/shr_staggered_product_grid_margin_top_second"` memberikan padding di bagian atas konten kartu sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`LinearLayout`: Ini adalah kontainer untuk mengatur tampilan dalam kartu produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="@dimen/shr_staggered_product_grid_card_width_landscape"` mengatur lebar `LinearLayout` sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:layout_height="wrap_content"` mengatur tinggi `LinearLayout` agar mengikuti konten yang ada di dalamnya.

`android:layout_marginEnd="@dimen/shr_staggered_product_grid_spacing_small"` dan

`android:layout_marginRight="@dimen/shr_staggered_product_grid_spacing_small"` memberikan margin di sisi kanan `LinearLayout` sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

`NetworkImageView`: Ini adalah komponen untuk menampilkan gambar produk dari jaringan. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_image"` memberikan ID kepada `ImageView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan
`android:layout_height="@dimen/shr_staggered_product_grid_card_height_landscape"` mengatur dimensi `ImageView` agar lebar mengisi parent dan tinggi sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:background="?attr/colorPrimaryDark"` menetapkan warna latar belakang `ImageView` menggunakan atribut warna dari tema aplikasi.

`android:scaleType="centerCrop"` mengatur jenis skala gambar agar selalu terpotong jika diperlukan untuk mengisi `ImageView` secara penuh.

`LinearLayout (Informasi Produk)`: Ini adalah kontainer untuk informasi detail produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `LinearLayout` agar lebar mengisi parent dan tinggi mengikuti konten di dalamnya.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

`android:padding="16dp"` memberikan padding di dalam `LinearLayout` agar konten terlihat lebih teratur.

TextView (Judul Produk dan Harga): Ini adalah komponen untuk menampilkan judul dan harga produk. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_title"` dan `android:id="@+id/product_price"` memberikan ID kepada TextView untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi TextView agar lebar mengisi parent dan tinggi mengikuti konten teks.

`android:text="@string/shr_product_title"` dan `android:text="@string/shr_product_description"` mengatur teks yang ditampilkan menggunakan string yang telah didefinisikan sebelumnya di dalam `strings.xml`.

`android:textAlignment="center"` mengatur teks agar berada di tengah secara horizontal.

File XML ini digunakan untuk menampilkan kartu produk dalam grid atau daftar dengan menggunakan `MaterialCardView`, yang merupakan pendekatan yang umum digunakan dalam pengembangan aplikasi Android untuk menampilkan item dengan desain yang konsisten dan modern.

Untuk `shr_staggered_product_card_third.xml`

Deklarasi XML: File dimulai dengan deklarasi XML standar yang menunjukkan versi XML dan encoding yang digunakan.

`MaterialCardView`: Ini adalah elemen utama dalam file XML yang digunakan untuk menampilkan kartu produk dengan tampilan yang konsisten dan elegan. Beberapa atribut yang digunakan di sini adalah:

`xmlns:android="http://schemas.android.com/apk/res/android"` dan `xmlns:app="http://schemas.android.com/apk/res-auto"` adalah namespace yang diperlukan untuk menggunakan atribut dan komponen Android serta Material Design.

`android:layout_width="@dimen/shr_staggered_product_grid_card_width_portrait"` dan `android:layout_height="match_parent"` mengatur dimensi `MaterialCardView` agar lebar sesuai dengan nilai yang didefinisikan di dalam `dimens.xml` (untuk lebar) dan tinggi mengisi seluruh ketinggian parent.

`app:cardBackgroundColor="@android:color/transparent"` menetapkan warna latar belakang kartu menjadi transparan.

`app:cardElevation="0dp"` menghilangkan elevasi bayangan dari kartu dengan mengatur nilai elevasi menjadi nol.

`app:cardPreventCornerOverlap="true"` mengatur agar sudut kartu tidak tumpang tindih satu sama lain.

`app:contentPaddingTop="@dimen/shr_staggered_product_grid_margin_top_third"` memberikan padding di bagian atas konten kartu sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

LinearLayout: Ini adalah kontainer untuk mengatur tampilan dalam kartu produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `LinearLayout` agar lebar mengisi parent dan tinggi mengikuti konten di dalamnya.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

NetworkImageView: Ini adalah komponen untuk menampilkan gambar produk dari jaringan. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_image"` memberikan ID kepada `ImageView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="@dimen/shr_staggered_product_grid_card_height_portrait"` mengatur dimensi `ImageView` agar lebar mengisi parent dan tinggi sesuai dengan nilai yang didefinisikan di dalam `dimens.xml`.

`android:background="?attr/colorPrimaryDark"` menetapkan warna latar belakang `ImageView` menggunakan atribut warna dari tema aplikasi.

`android:scaleType="centerCrop"` mengatur jenis skala gambar agar selalu terpotong jika diperlukan untuk mengisi `ImageView` secara penuh.

LinearLayout (Informasi Produk): Ini adalah kontainer untuk informasi detail produk. Beberapa atribut yang digunakan di sini adalah:

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `LinearLayout` agar lebar mengisi parent dan tinggi mengikuti konten di dalamnya.

`android:orientation="vertical"` mengatur orientasi kontainer menjadi vertikal, sehingga elemen-elemen di dalamnya disusun dari atas ke bawah.

`android:padding="16dp"` memberikan padding di dalam `LinearLayout` agar konten terlihat lebih teratur.

`TextView` (Judul Produk dan Harga): Ini adalah komponen untuk menampilkan judul dan harga produk. Beberapa atribut yang digunakan di sini adalah:

`android:id="@+id/product_title"` dan `android:id="@+id/product_price"` memberikan ID kepada `TextView` untuk referensi di kode Kotlin.

`android:layout_width="match_parent"` dan `android:layout_height="wrap_content"` mengatur dimensi `TextView` agar lebar mengisi parent dan tinggi mengikuti konten teks.

`android:text="@string/shr_product_title"` dan `android:text="@string/shr_product_description"` mengatur teks yang ditampilkan menggunakan string yang telah didefinisikan sebelumnya di dalam `strings.xml`.

`android:textAlignment="center"` mengatur teks agar berada di tengah secara horizontal.

File XML ini digunakan untuk menampilkan kartu produk dalam mode potret (portrait) menggunakan `MaterialCardView`, yang menyediakan tampilan yang konsisten dan modern untuk item-item produk dalam aplikasi Android.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/farlyhaydyhdjalil/Praktikum-PEMROGRAMANMOBILE/tree/33d1bab715da094dfa0ef596713879410f9e60cd/Modul%205/shrineApp>