**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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Farlyhaydy H.Djalil

NIM : 2210817210006

|  |  |
| --- | --- |
| Menyetujui,  Asisten Praktikum  Muhammad Asandy Putra  NIM. 2110817310002 | Mengetahui,  Dosen Penanggung Jawab Praktikum  Muti’a Maulida, S.Kom., M.T.I.  NIP. 198810272019032013 |

# DAFTAR ISI

[LEMBAR PENGESAHAN 2](#_Toc170569258)

[DAFTAR ISI 3](#_Toc170569259)

[DAFTAR GAMBAR 5](#_Toc170569260)

[DAFTAR TABEL 7](#_Toc170569261)

[MODUL 1 : ANDROID BASIC WITH KOTLIN 9](#_Toc170569262)

[SOAL 1 9](#_Toc170569263)

[A. Source Code 11](#_Toc170569264)

[B. Output Program 16](#_Toc170569265)

[C. Pembahasan 18](#_Toc170569266)

[D. Tautan Git 25](#_Toc170569267)

[MODUL 2 : ANDROID LAYOUT 27](#_Toc170569268)

[SOAL 1 27](#_Toc170569269)

[A. Source Code 28](#_Toc170569270)

[B. Output Program 33](#_Toc170569271)

[C. Pembahasan 35](#_Toc170569272)

[D. Tautan Git 44](#_Toc170569273)

[MODUL 3 : ANDROID NAVIGATION 45](#_Toc170569274)

[SOAL 1 45](#_Toc170569275)

[A. Source Code 47](#_Toc170569276)

[B. Output Program 61](#_Toc170569277)

[C. Pembahasan 63](#_Toc170569278)

[D. Tautan Git 85](#_Toc170569279)

[MODUL 4 : CONNECT TO THE INTERNET 86](#_Toc170569280)

[SOAL 1 86](#_Toc170569281)

[A. Source Code 88](#_Toc170569282)

[B. Output Program 107](#_Toc170569283)

[C. Pembahasan 108](#_Toc170569284)

[D. Tautan Git 119](#_Toc170569285)

[MODUL 5 : ANDROID UI DESIGN 120](#_Toc170569286)

[SOAL 1 120](#_Toc170569287)

[A. Source Code 121](#_Toc170569288)

[B. Output Program 152](#_Toc170569289)

[C. Pembahasan 154](#_Toc170569290)

[D. Tautan Git 181](#_Toc170569291)

# DAFTAR GAMBAR

**Modul 1 : Android Basic with Kotlin**

[Gambar 1. Tampilan Awal Aplikasi 9](#_Toc170517116)

[Gambar 2. Tampilan Dadu Setelah Di Roll 10](#_Toc170517117)

[Gambar 3. Tampilan Roll Dadu Double 11](#_Toc170517118)

[Gambar 4. Screenshot Hasil Jawaban Soal 1 yang ke 1 16](#_Toc170517119)

[Gambar 5. Screenshot Hasil Jawaban Soal 1 yang ke 2 17](#_Toc170517120)

[Gambar 6. Screenshot Hasil Jawaban Soal 1 yang ke 3 18](#_Toc170517121)

**Modul 2 : Android Layout**

[Gambar 7. Tampilan Awal Aplikasi 27](#_Toc170517122)

[Gambar 8. Tampilan Aplikasi Setelah Dijalankan 28](#_Toc170517123)

[Gambar 9. Screenshot Hasil Jawaban Soal 1 yang ke 1 33](#_Toc170517124)

[Gambar 10. Screenshot Hasil Jawaban Soal 1 yang ke 2 34](#_Toc170517125)

**Modul 3 : Android Navigation**

[Gambar 11. Contoh pertama 45](#_Toc170517126)

[Gambar 12. Contoh kedua 46](#_Toc170517127)

[Gambar 13. Contoh Ketiga 47](#_Toc170517128)

[Gambar 14. Screenshot Hasil Jawaban Soal 1 yang ke 1 61](#_Toc170517129)

[Gambar 15. Screenshot Hasil Jawaban Soal 1 yang ke 2 62](#_Toc170517130)

[Gambar 16. Screenshot Hasil Jawaban Soal 1 yang ke 3 63](#_Toc170517131)

**Modul 4 : Connect to the internet**

[Gambar 17. Contoh gambar pertama 87](#_Toc170517132)

[Gambar 18. Contoh gambar kedua 87](#_Toc170517133)

[Gambar 19. Screenshot Hasil Jawaban Soal 1 yang ke 1 107](#_Toc170517134)

[Gambar 20. Screenshot Hasil Jawaban Soal 1 yang ke 2 108](#_Toc170517135)

**Modul 5 : Android UI Design**

[Gambar 21. Contoh gambar pertama 120](#_Toc170517136)

[Gambar 22. Contoh gambar kedua 121](#_Toc170517137)

[Gambar 23. Screenshot Hasil Jawaban Soal 1 tampilan login 152](#_Toc170517138)

[Gambar 24. Screenshot Hasil Jawaban Soal 1 tampilan di halaman utama 153](#_Toc170517139)

[Gambar 25. Screenshot Hasil Jawban Soal 1 tampilan menu 154](#_Toc170517140)

# DAFTAR TABEL

**Modul 1 : Android Basic with Kotlin**

[Tabel 1. Source Code Soal 1 yang Kotlin 14](#_Toc170517141)

[Tabel 2. Source Code Soal 1 yang XML 15](#_Toc170517142)

**Modul 2 : Android Layout**

[Tabel 3. Source Code Soal 1 yang Kotlin 29](#_Toc170517143)

[Tabel 4. Source Code Soal 1 yang XML 32](#_Toc170517144)

**Modul 3 : Android Navigation**

[Tabel 5. Source Code Soal 1 Kotlin yang pertama 50](#_Toc170517145)

[Tabel 6. Source Code Soal 1 Kotlin yang kedua 52](#_Toc170517146)

[Tabel 7. Source Code Soal 1 Kotlin yang ketiga 54](#_Toc170517147)

[Tabel 8. Source Code Soal 1 Kotlin yang keempat 55](#_Toc170517148)

[Tabel 9. Source Code Soal 1 XML yang pertama 59](#_Toc170517149)

[Tabel 10. Source Code Soal 1 XML yang kedua 60](#_Toc170517150)

**Modul 4 : Connect to the internet**

[Tabel 11. Source Code Soal 1 Kotlin yang pertama 89](#_Toc170517151)

[Tabel 12. Source Code Soal 1 Kotlin yang kedua 91](#_Toc170517152)

[Tabel 13. Source Code Soal 1 Kotlin yang ketiga 92](#_Toc170517153)

[Tabel 14. Source Code Soal 1 Kotlin yang keempat 95](#_Toc170517154)

[Tabel 15. Source Code Soal 1 Kotlin yang kelima 96](#_Toc170517155)

[Tabel 16. Source Code Soal 1 Kotlin yang keenam 99](#_Toc170517156)

[Tabel 17. Source Code Soal 1 Kotlin yang ketujuh 100](#_Toc170517157)

[Tabel 18. Source Code Soal 1 XML yang pertama 101](#_Toc170517158)

[Tabel 19. Source Code Soal 1 XML yang kedua 103](#_Toc170517159)

[Tabel 20. Source Code Soal 1 XML yang ketiga 105](#_Toc170517160)

[Tabel 21. Source Code Soal 1 XML yang keempat 106](#_Toc170517161)

**Modul 5 : Android UI Design**

[Tabel 22. Source Code Soal 1 Kotlin yang pertama 122](#_Toc170517162)

[Tabel 23. Source Code Soal 1 Kotlin yang kedua 123](#_Toc170517163)

[Tabel 24. Source Code Soal 1 Kotlin yang ketiga 124](#_Toc170517164)

[Tabel 25. Source Code Soal 1 Kotlin yang keempat 126](#_Toc170517165)

[Tabel 26. Source Code Soal 1 Kotlin yang kelima 127](#_Toc170517166)

[Tabel 27. Source Code Soal 1 Kotlin yang keenam 129](#_Toc170517167)

[Tabel 28. Source Code Soal 1 Kotlin yang ketujuh 130](#_Toc170517168)

[Tabel 29. Source Code Soal 1 Kotlin yang kedelapan 130](#_Toc170517169)

[Tabel 30. Source Code Soal 1 Kotlin yang kesembilan 132](#_Toc170517170)

[Tabel 31. Source Code Soal 1 Kotlin yang kesepuluh 133](#_Toc170517171)

[Tabel 32. Source Code Soal 1 Kotlin yang kesebelas 134](#_Toc170517172)

[Tabel 33. Source Code Soal 1 Kotlin yang kedua belas 137](#_Toc170517173)

[Tabel 34. Source Code Soal 1 Kotlin yang ketiga belas 137](#_Toc170517174)

[Tabel 35. Source Code Soal 1 XML yang pertama 139](#_Toc170517175)

[Tabel 36. Source Code Soal 1 XML yang kedua 142](#_Toc170517176)

[Tabel 37. Source Code Soal 1 XML yang ketiga 142](#_Toc170517177)

[Tabel 38. Source Code Soal 1 XML yang keempat 144](#_Toc170517178)

[Tabel 39. Source Code Soal 1 XML yang kelima 146](#_Toc170517179)

[Tabel 40. Source Code Soal 1 XML yang keenam 147](#_Toc170517180)

[Tabel 41. Source Code Soal 1 XML yang ketujuh 149](#_Toc170517181)

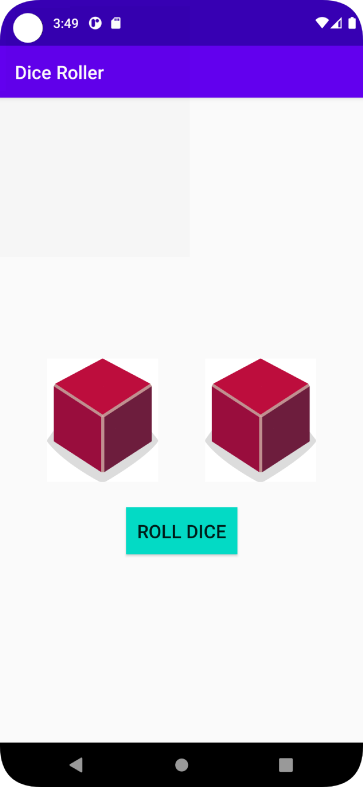
[Tabel 42. Source Code Soal 1 XML yang kedelepan 151](#_Toc170517182)

# 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:

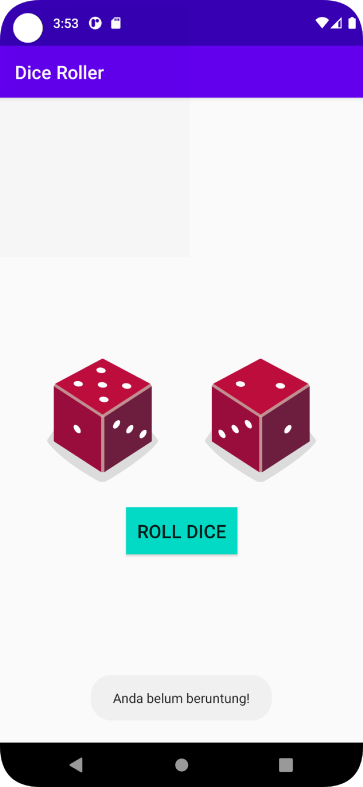
1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



Gambar 1. Tampilan Awal Aplikasi

**Gambar 1 Tampilan Awal Aplikasi**

1. 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.

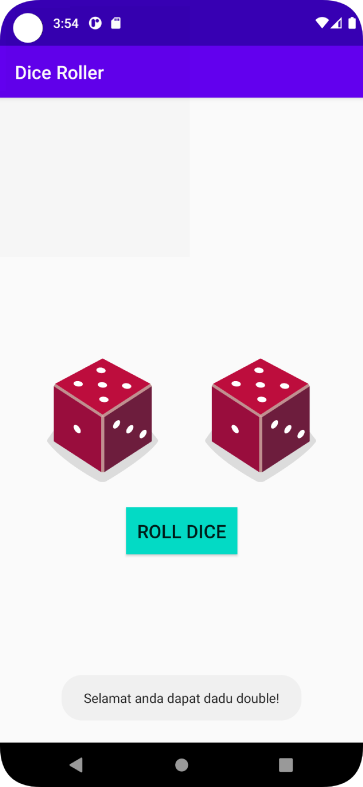


Gambar 2. Tampilan Dadu Setelah Di Roll

**Gambar 2 Tampilan Dadu Setelah Di Roll**

1. 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.
2. 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.
3. Untuk gambar dadu dapat didownload pada link berikut:

[https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N\_5OMW81Ll&export= download](https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export=%20download%20%20)



Gambar 3. Tampilan Roll Dadu Double

**Gambar 3 Tampilan Roll Dadu Double**

### A. Source Code

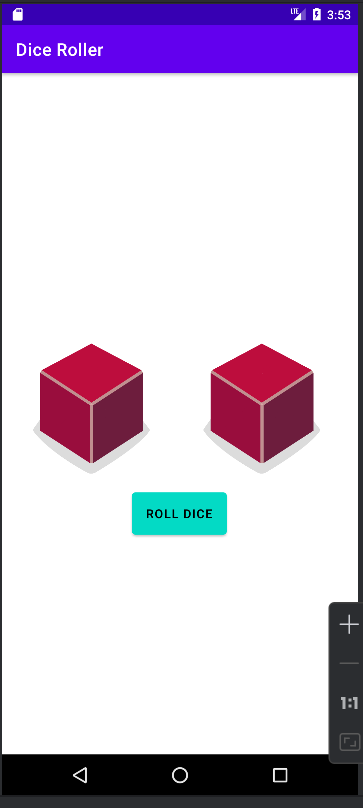
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | package com.example.RoleDice  import android.graphics.Color import android.graphics.PorterDuff import android.widget.TextView import android.widget.Toast import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Button import android.widget.ImageView import com.example.tugasroledice.R  class MainActivity : AppCompatActivity() {   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_main*)   val rollButton: Button = findViewById(R.id.*let\_roll*)  rollButton.setOnClickListener **{** roll() **}** }   private fun roll() {  var dadulagi1 = diceroll1()  var dadulagi2 = diceroll2()   if (dadulagi1 == dadulagi2) {  showCustomToast("Selamat anda dapat dadu double!", Color.parseColor("#F0F0F0"), Color.*BLACK*)  } else {  showCustomToast("Anda belum beruntung!", Color.parseColor("#F0F0F0"), Color.*BLACK*)  }  }   private fun showCustomToast(message: String, backgroundColor: Int, textColor: Int) {  val toast = Toast.makeText(this, message, Toast.*LENGTH\_SHORT*)  val view = toast.*view* // Mengatur warna latar belakang Toast  view?.*background*?.setColorFilter(backgroundColor, PorterDuff.Mode.*SRC\_IN*)   // Mendapatkan TextView dari Toast sehingga dapat diedit  val text = view?.findViewById<TextView>(android.R.id.*message*)  text?.setTextColor(textColor)   toast.show()  }   private fun diceroll1(): Int {  val dice1 = Dice(6)  val diceRoll1 = dice1.roll()   val diceImage1: ImageView = findViewById(R.id.*dice\_image1*)  val drawableResource1 = when (diceRoll1) {  1 -> R.drawable.*dice\_1* 2 -> R.drawable.*dice\_2* 3 -> R.drawable.*dice\_3* 4 -> R.drawable.*dice\_4* 5 -> R.drawable.*dice\_5* else -> R.drawable.*dice\_6* }   diceImage1.setImageResource(drawableResource1)  diceImage1.*contentDescription* = diceRoll1.toString()  return diceRoll1  }   private fun diceroll2(): Int {  val dice2 = Dice(6)  val diceRoll2 = dice2.roll()   val diceImage2: ImageView = findViewById(R.id.*dice\_image2*)  val drawableResource2 = when (diceRoll2) {  1 -> R.drawable.*dice\_1* 2 -> R.drawable.*dice\_2* 3 -> R.drawable.*dice\_3* 4 -> R.drawable.*dice\_4* 5 -> R.drawable.*dice\_5* else -> R.drawable.*dice\_6* }   diceImage2.setImageResource(drawableResource2)  diceImage2.*contentDescription* = diceRoll2.toString()  return diceRoll2  } }  class Dice(private val numSides: Int) {   */\*\*  \* Lakukan pelemparan dadu secara acak dan mengembalikan hasilnya.  \*/* fun roll(): Int {  return (1..numSides).*random*()  } } |

Tabel 1. Source Code Soal 1 yang Kotlin

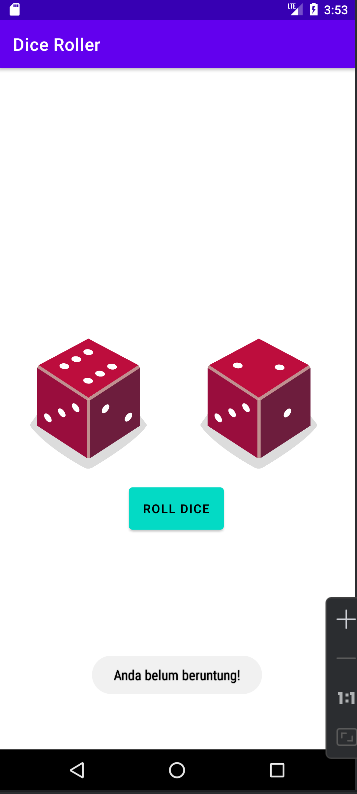
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout 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"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_gravity="center\_vertical"  android:orientation="vertical"  tools:context="com.example.RoleDice.MainActivity">   <ImageView  android:id="@+id/dice\_image1"  android:layout\_width="150dp"  android:layout\_height="150dp"  android:layout\_marginStart="28dp"  android:layout\_marginTop="312dp"  android:src="@drawable/empty\_dice"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />   <ImageView  android:id="@+id/dice\_image2"  android:layout\_width="150dp"  android:layout\_height="150dp"  android:layout\_marginTop="312dp"  android:layout\_marginEnd="36dp"  android:src="@drawable/empty\_dice"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />   <Button  android:id="@+id/let\_roll"  android:layout\_width="wrap\_content"  android:layout\_height="60dp"  android:layout\_marginTop="16dp"  android:backgroundTint="#FF03DAC5"  android:text="@string/roll\_dice"  android:textColor="#000000"  app:layout\_constraintBaseline\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintHorizontal\_bias="0.498"    app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/dice\_image1" />  </androidx.constraintlayout.widget.ConstraintLayout> |

Tabel 2. Source Code Soal 1 yang XML

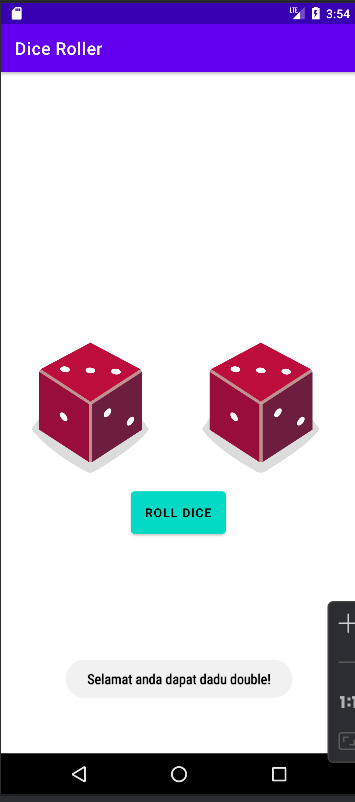
### B. Output Program



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 menetapkannya 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 menetapkannya 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 menetapkannya 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 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.yout.

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/774512e4ffff5969d626743ab171dc40fd6ee091/Modul%201/Dice%20Role>

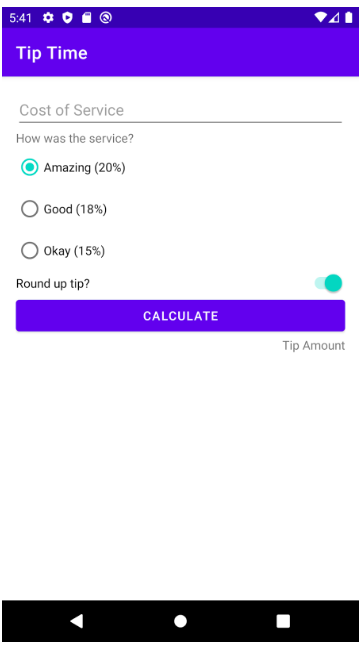
# MODUL 2 : ANDROID LAYOUT

## SOAL 1

**Soal Praktikum:**

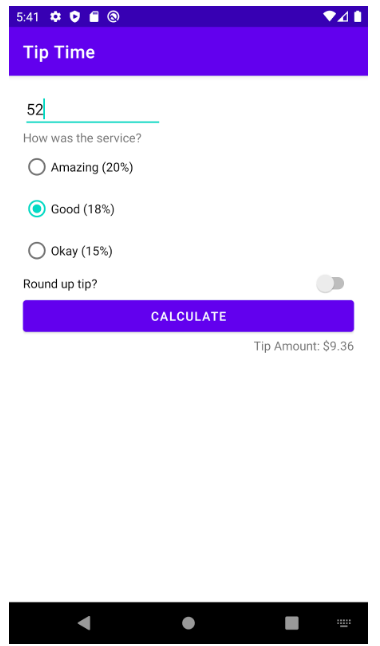
1. Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

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

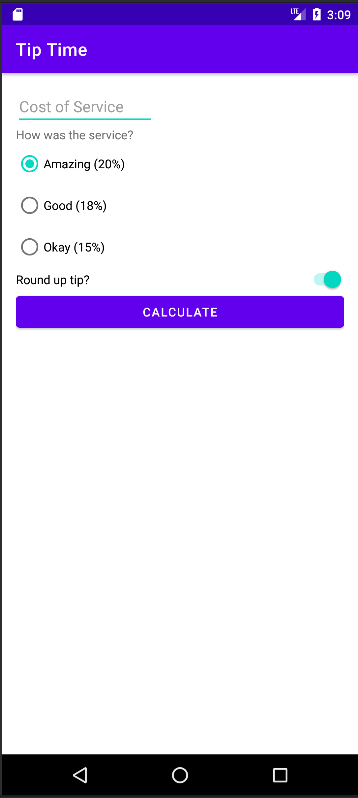
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | package com.example.tipcalculatorapp  import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import com.example.tipcalculatorapp.databinding.ActivityMainBinding import java.text.NumberFormat  class MainActivity : AppCompatActivity() {   private lateinit var binding: ActivityMainBinding   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   binding = ActivityMainBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)   binding.calculateButton.setOnClickListener **{** calculateTip() **}** }   private fun calculateTip() {  val stringInTextField = binding.costOfService.*text*.toString()  val cost = stringInTextField.*toDoubleOrNull*()  if (cost == null) {  binding.tipResult.*text* = ""  return  }   val tipPercentage = when (binding.tipOptions.*checkedRadioButtonId*) {  R.id.*option\_twenty\_percent* -> 0.20  R.id.*option\_eighteen\_percent* -> 0.18  else -> 0.15  }   var tip = tipPercentage \* cost  if (binding.roundUpSwitch.*isChecked*) {  tip = kotlin.math.*ceil*(tip)  }   val formattedTip = NumberFormat.getCurrencyInstance().format(tip)  binding.tipResult.*text* = getString(R.string.*tip\_amount*, formattedTip)  } } |

Tabel 3. Source Code Soal 1 yang Kotlin

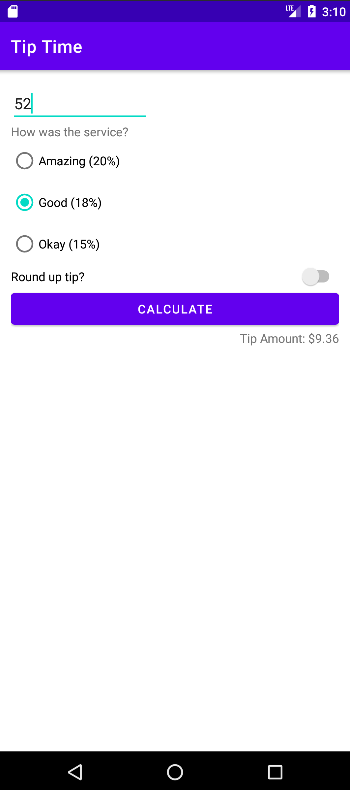
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 | <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout  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"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:padding="16dp"  tools:context=".MainActivity">   <EditText  android:id="@+id/cost\_of\_service"  android:layout\_width="160dp"  android:layout\_height="wrap\_content"  android:hint="@string/cost\_of\_service"  android:inputType="numberDecimal"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent" />   <TextView  android:id="@+id/service\_question"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/how\_was\_the\_service"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/cost\_of\_service" />   <RadioGroup  android:id="@+id/tip\_options"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:checkedButton="@id/option\_twenty\_percent"  android:orientation="vertical"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/service\_question">   <RadioButton  android:id="@+id/option\_twenty\_percent"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/amazing\_service" />   <RadioButton  android:id="@+id/option\_eighteen\_percent"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/good\_service" />   <RadioButton  android:id="@+id/option\_fifteen\_percent"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/ok\_service" />  </RadioGroup>   <Switch  android:id="@+id/round\_up\_switch"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:checked="true"  android:text="@string/round\_up\_tip"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="@id/tip\_options"  app:layout\_constraintTop\_toBottomOf="@id/tip\_options"  tools:ignore="UseSwitchCompatOrMaterialXml" />   <Button  android:id="@+id/calculate\_button"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:text="@string/calculate"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/round\_up\_switch" />   <TextView  android:id="@+id/tip\_result"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  tools:text="Tip Amount: $10"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintTop\_toBottomOf="@id/calculate\_button" /> </androidx.constraintlayout.widget.ConstraintLayout> |

Tabel 4. Source Code Soal 1 yang XML

### B. Output Program



Gambar 9. Screenshot Hasil Jawaban Soal 1 yang ke 1



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, private 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 mengembang (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/e93522abdeecee3e91dc975485016e7f50afcd4d/Modul%202/Calculator%20Tip>

# MODUL 3 : ANDROID NAVIGATION

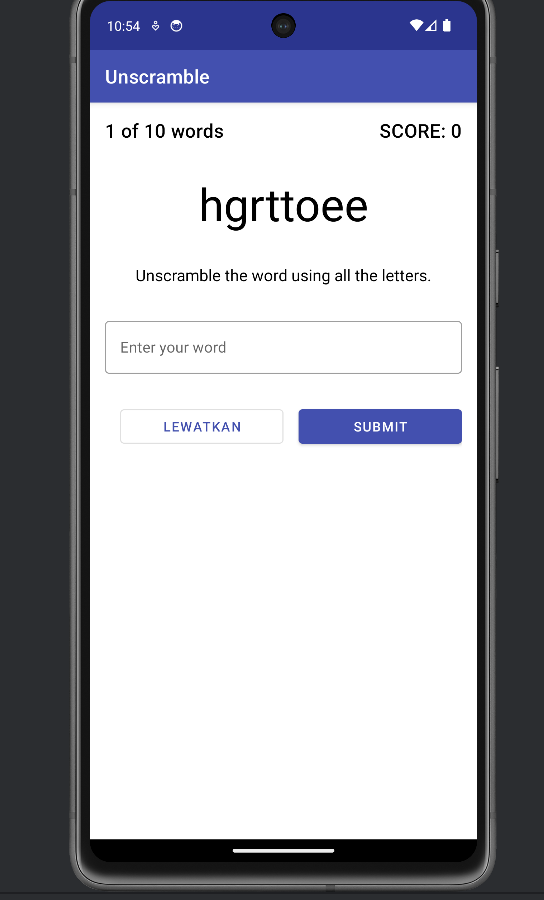
## SOAL 1

**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.

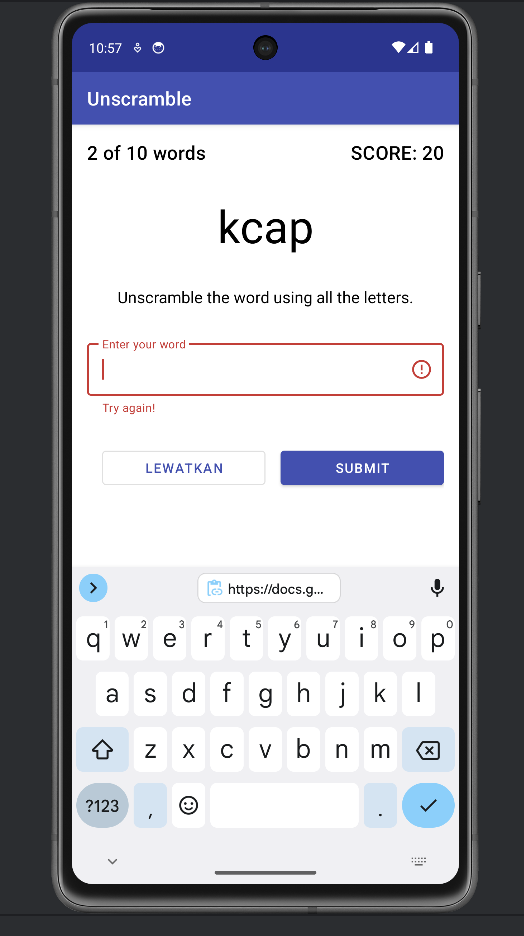
Contoh Hasil Aplikasi :

1. Saat menjawab quiz



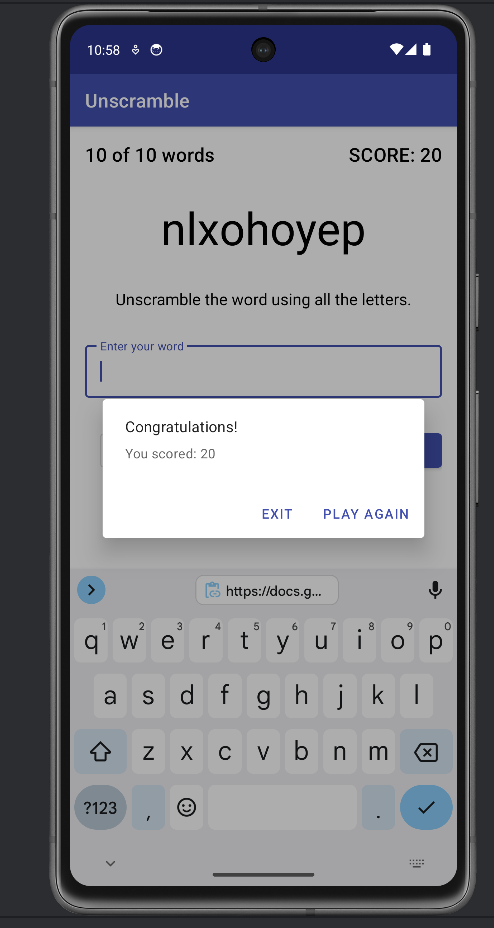
Gambar 11. Contoh pertama

1. Saat field input dibiarkan kosong dan disubmit



Gambar 12. Contoh kedua

1. Menampilkan hasil score akhir setelah 10 soal terjawab



Gambar 13. Contoh Ketiga

### A. Source Code

**GameFragment.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94 | package com.example.android.unscramble.ui.game  import android.os.Bundle import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import androidx.databinding.DataBindingUtil import androidx.fragment.app.Fragment import androidx.fragment.app.viewModels import com.example.android.unscramble.R import com.example.android.unscramble.databinding.GameFragmentBinding import com.google.android.material.dialog.MaterialAlertDialogBuilder  class GameFragment : Fragment() {   private lateinit var binding: GameFragmentBinding   private val viewModel: GameViewModel by *viewModels*()   override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?  ): View {   binding = DataBindingUtil.inflate(inflater, R.layout.*game\_fragment*, container, false)  return binding.*root* }   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  super.onViewCreated(view, savedInstanceState)   binding.*gameViewModel* = viewModel  binding.*maxNoOfWords* = *MAX\_NO\_OF\_WORDS* binding.*lifecycleOwner* = *viewLifecycleOwner* binding.submit.setOnClickListener **{** onSubmitWord() **}** binding.skip.setOnClickListener **{** onSkipWord() **}** }   private fun onSubmitWord() {  val playerWord = binding.textInputEditText.*text*.*toString*()   if (viewModel.isUserWordCorrect(playerWord)) {  setErrorTextField(false)  if (!viewModel.nextWord()) {  showFinalScoreDialog()  }  } else {  setErrorTextField(true)  }  }   private fun onSkipWord() {  if (viewModel.nextWord()) {  setErrorTextField(false)  } else {  showFinalScoreDialog()  }  }   private fun showFinalScoreDialog() {  MaterialAlertDialogBuilder(requireContext())  .setTitle(getString(R.string.*congratulations*))  .setMessage(getString(R.string.*you\_scored*, viewModel.score.*value*))  .setCancelable(false)  .setNegativeButton(getString(R.string.*exit*)) **{** \_, \_ **->** exitGame()  **}** .setPositiveButton(getString(R.string.*play\_again*)) **{** \_, \_ **->** restartGame()  **}** .show()  }   private fun restartGame() {  viewModel.reinitializeData()  setErrorTextField(false)  }   private fun exitGame() {  *activity*?.finish()  }   private fun setErrorTextField(error: Boolean) {  if (error) {  binding.textField.*isErrorEnabled* = true  binding.textField.*error* = getString(R.string.*try\_again*)  } else {  binding.textField.*isErrorEnabled* = false  binding.textInputEditText.*text* = null  }  } } |

Tabel 5. Source Code Soal 1 Kotlin yang pertama

**GameViewModel.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88 | package com.example.android.unscramble.ui.game  import android.text.Spannable import android.text.SpannableString import android.text.style.TtsSpan import android.util.Log import androidx.lifecycle.LiveData import androidx.lifecycle.MutableLiveData import androidx.lifecycle.Transformations import androidx.lifecycle.ViewModel  class GameViewModel : ViewModel() {  private val \_score = MutableLiveData(0)  val score: LiveData<Int>  get() = \_score   private val \_currentWordCount = MutableLiveData(0)  val currentWordCount: LiveData<Int>  get() = \_currentWordCount   private val \_currentScrambledWord = MutableLiveData<String>()  val currentScrambledWord: LiveData<Spannable> = Transformations.map(\_currentScrambledWord) **{** if (**it** == null) {  SpannableString("")  } else {  val scrambledWord = **it**.toString()  val spannable: Spannable = SpannableString(scrambledWord)  spannable.setSpan(  TtsSpan.VerbatimBuilder(scrambledWord).build(),  0,  scrambledWord.length,  Spannable.*SPAN\_INCLUSIVE\_INCLUSIVE* )  spannable  }  **}** private var wordsList: MutableList<String> = *mutableListOf*()  private lateinit var currentWord: String   init {  getNextWord()  }   private fun getNextWord() {  currentWord = *allWordsList*.*random*()  val tempWord = currentWord.*toCharArray*()  tempWord.*shuffle*()   while (*String*(tempWord).*equals*(currentWord, false)) {  tempWord.*shuffle*()  }  if (wordsList.contains(currentWord)) {  getNextWord()  } else {  Log.d("Unscramble", "currentWord= $currentWord")  \_currentScrambledWord.*value* = *String*(tempWord)  \_currentWordCount.*value* = \_currentWordCount.*value*?.inc()  wordsList.add(currentWord)  }  }   fun reinitializeData() {  \_score.*value* = 0  \_currentWordCount.*value* = 0  wordsList.clear()  getNextWord()  }   private fun increaseScore() {  \_score.*value* = \_score.*value*?.plus(*SCORE\_INCREASE*)  }   fun isUserWordCorrect(playerWord: String): Boolean {  if (playerWord.*equals*(currentWord, true)) {  increaseScore()  return true  }  return false  }   fun nextWord(): Boolean {  return if (\_currentWordCount.*value*!! < *MAX\_NO\_OF\_WORDS*) {  getNextWord()  true  } else false  } } |

Tabel 6. Source Code Soal 1 Kotlin yang kedua

**ListofWords.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package com.example.android.unscramble.ui.game   const val *MAX\_NO\_OF\_WORDS* = 10 const val *SCORE\_INCREASE* = 20  // List with all the words for the Game val *allWordsList*: List<String> =  *listOf*("apel", "angin", "anak", "asap", "asri", "awan", "ayam", "ajar", "amal", "adab",  "air", "aku", "ayam", "api", "aduh", "anjing", "antar", "arah", "abadi", "alun",  "buku", "bola", "ban", "batu", "baik", "buah", "baju", "bangku", "bunga", "belajar",  "bambu", "besar", "belut", "berita", "burung", "cinta", "cawan", "cari", "cahaya", "cat",  "cuka", "cacing", "ceri", "celana", "cerah", "celana", "cabang", "cetak", "cerdas", "cukur",  "daun", "dasi", "daging", "dapur", "dada", "dekat", "diri", "domba", "duduk", "darat",  "durian", "dalam", "damai", "danau", "dosen", "elang", "empat", "es", "emas", "enak",  "empat", "elok", "ekor", "epal", "embun", "fajar", "film", "foto", "fasilitas", "faktor",  "fiksi", "final", "firasat", "fantasi", "flora", "furnitur", "firman", "fisika", "formal", "fitur",  "gelas", "garam", "gula", "gajah", "gunung", "gembira", "guntur", "gabus", "gadis", "gajah",  "gagang", "gantung", "gantung", "gas", "gerak", "hujan", "hutan", "hari", "hati", "hewan",  "hidup", "harum", "hotel", "hitam", "halaman", "ikan", "indah", "ilmu", "istana", "inspirasi",  "jalan", "jendela", "jam", "jatuh", "jari", "jambu", "jaring", "jernih", "jeruk", "jawaban",  "kucing", "kuda", "kursi", "kertas", "kapal", "kota", "kaki", "kamar", "kasur",  "kayak", "langit", "laut", "lilin", "lembut", "lentera", "lintas", "layar", "lebar", "lembah", "lampu",  "matahari", "meja", "makan", "minum", "manis","mimpi", "mobil", "muda", "mudah", "musik",  "nasi", "nyamuk", "nangka", "naik", "nyata", "nyanyi", "nona", "novel", "namun", "nara",  "ombak", "oren", "obyek", "oleh", "otak", "otot", "ongkos", "opini", "olah", "obat",  "pohon", "pisau", "pantai", "padi", "pagar", "payung", "piring", "pulau", "pusat", "panas",  "papan", "pintu", "pedas", "pelangi", "pena", "quran", "quadrat", "quark", "quasar", "quiz",  "robot", "rumah", "rusa", "roda", "rapi", "rak", "rawat", "rasa", "rakit", "rambut",  "sawah", "sapi", "sinar", "susu", "sawah", "sepeda", "surat", "sore", "sopan", "sarung",  "taman", "tanah", "tanda", "tangan", "tali", "tahu", "takut", "tas", "tungku", "topi",  "ular", "udara", "ubi", "ulang", "umur", "upah", "udang", "usaha", "utama", "uang",  "vaksin", "vila", "vokal", "valuta", "vital", "venus", "varian", "virus", "vitamin", "volume",  "waktu", "warna", "wajah", "waktu", "wangi", "warung", "wadah", "wawasan", "wacana", "waspada",  "xilofon", "xenon", "xilografi", "xerofit", "xenofobia", "yoghurt", "yak", "yodium", "yoyo", "yoga",  "yakin", "yuk", "yatim", "yul", "yumpie", "zebra", "zaman", "zakat", "zodiak", "zona") |

Tabel 7. Source Code Soal 1 Kotlin yang ketiga

**MainActivity.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package com.example.android.unscramble  import android.os.Bundle import androidx.appcompat.app.AppCompatActivity  */\*\*  \* Creates an Activity that hosts the Game fragment in the app  \*/* class MainActivity : AppCompatActivity() {   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*main\_activity*)  } } |

Tabel 8. Source Code Soal 1 Kotlin yang keempat

**game\_fragment.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119 | <?xml version="1.0" encoding="utf-8"?>  <layout 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">   <data>  <variable  name="gameViewModel"  type="com.example.android.unscramble.ui.game.GameViewModel" />  <variable  name="maxNoOfWords"  type="int" />  </data>   <ScrollView  android:layout\_width="match\_parent"  android:layout\_height="match\_parent">   <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:padding="@dimen/default\_padding"  tools:context=".ui.game.GameFragment">   <Button  android:id="@+id/skip"  style="?attr/materialButtonOutlinedStyle"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_marginStart="@dimen/default\_padding"  android:layout\_marginEnd="@dimen/default\_padding"  android:text="@string/skip"  app:layout\_constraintBaseline\_toBaselineOf="@+id/submit"  app:layout\_constraintEnd\_toStartOf="@+id/submit"  app:layout\_constraintStart\_toStartOf="parent" />   <Button  android:id="@+id/submit"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_marginTop="@dimen/default\_margin"  android:text="@string/submit"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toEndOf="@+id/skip"  app:layout\_constraintTop\_toBottomOf="@+id/textField" />   <TextView  android:id="@+id/textView\_instructions"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/instructions"  android:textSize="17sp"  app:layout\_constraintBottom\_toTopOf="@+id/textField"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/textView\_unscrambled\_word" />   <TextView  android:id="@+id/textView\_unscrambled\_word"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="@dimen/default\_margin"  android:layout\_marginBottom="@dimen/default\_margin"  android:text="@{gameViewModel.currentScrambledWord}"  android:textAppearance="@style/TextAppearance.MaterialComponents.Headline3"  app:layout\_constraintBottom\_toTopOf="@+id/textView\_instructions"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/word\_count"  tools:text="Scramble word" />   <TextView  android:id="@+id/word\_count"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@{@string/word\_count(gameViewModel.currentWordCount, maxNoOfWords)}"  android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"  app:layout\_constraintBottom\_toTopOf="@+id/textView\_unscrambled\_word"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  tools:text="3 of 10 words" />   <TextView  android:id="@+id/score"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@{@string/score(gameViewModel.score)}"  android:textAllCaps="true"  android:textAppearance="@style/TextAppearance.MaterialComponents.Headline6"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  tools:text="Score: 20" />   <com.google.android.material.textfield.TextInputLayout  android:id="@+id/textField"  style="@style/Widget.Unscramble.TextInputLayout.OutlinedBox"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_marginTop="@dimen/default\_margin"  android:hint="@string/enter\_your\_word"  app:errorIconDrawable="@drawable/ic\_error"  app:helperTextTextAppearance="@style/TextAppearance.MaterialComponents.Subtitle1"  app:layout\_constraintBottom\_toTopOf="@+id/submit"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/textView\_instructions">   <com.google.android.material.textfield.TextInputEditText  android:id="@+id/text\_input\_edit\_text"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:inputType="textPersonName|textNoSuggestions"  android:maxLines="1" />  </com.google.android.material.textfield.TextInputLayout>   </androidx.constraintlayout.widget.ConstraintLayout>  </ScrollView> </layout> |

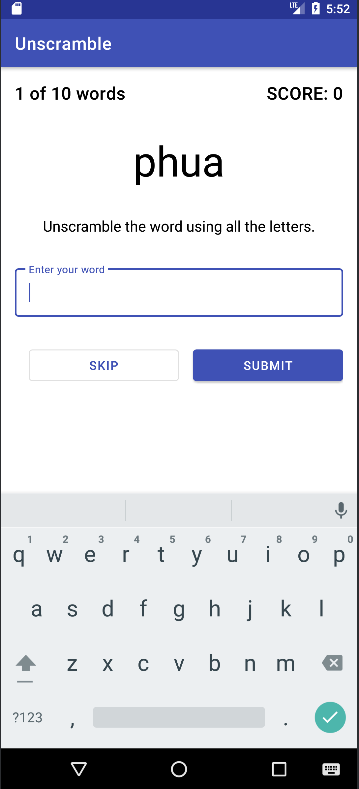
Tabel 9. Source Code Soal 1 XML yang pertama

**main\_activity.xml**

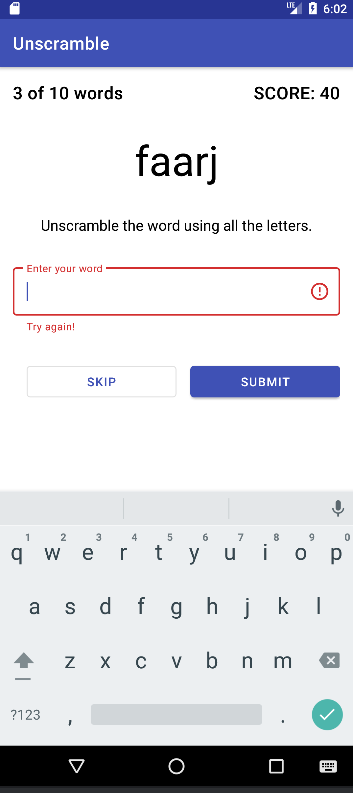
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | <?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".MainActivity">   <androidx.fragment.app.FragmentContainerView  android:id="@+id/game\_fragment"  android:name="com.example.android.unscramble.ui.game.GameFragment"  android:layout\_width="0dp"  android:layout\_height="0dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent" /> </androidx.constraintlayout.widget.ConstraintLayout> |

Tabel 10. Source Code Soal 1 XML yang kedua

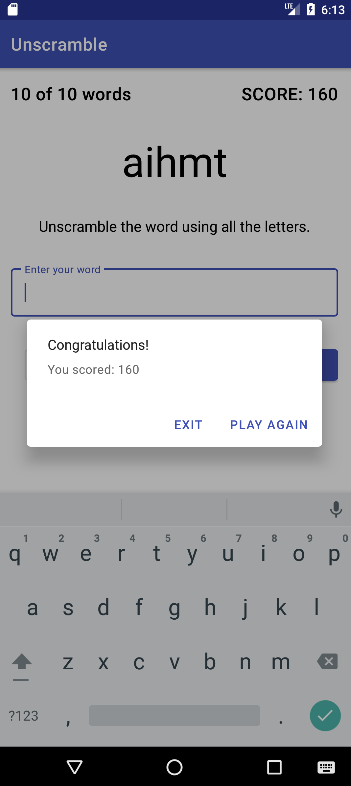
### B. Output Program



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 mengembang (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.isErrorEnabled = 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.isErrorEnabled = 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, adalan 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:**

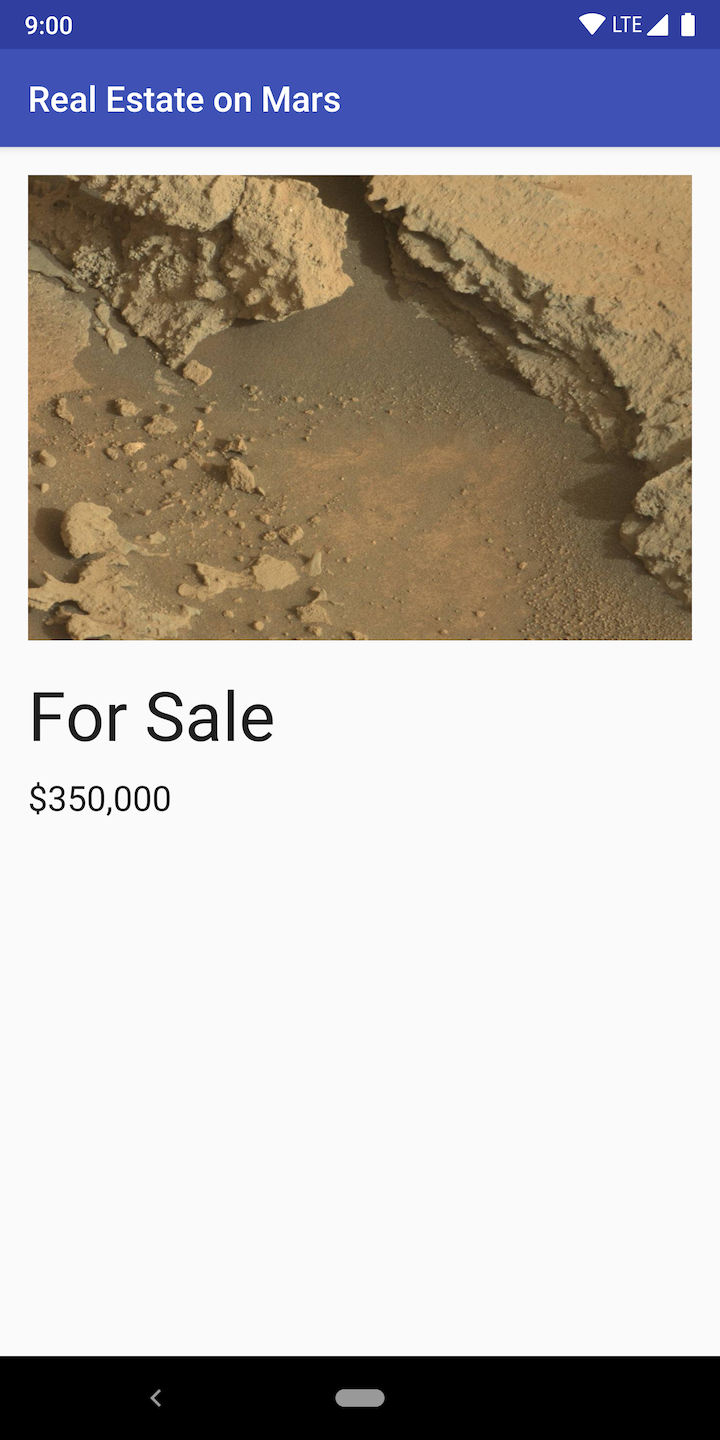
1. Buatlah sebuah aplikasi Android sederhana untuk menampilkan data dari Internet melalui Public API

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  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*/  package com.example.android.marsrealestate.detail  import android.os.Bundle import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import androidx.fragment.app.Fragment import androidx.lifecycle.ViewModelProvider import com.example.android.marsrealestate.databinding.FragmentDetailBinding  */\*\*  \* This [Fragment] shows the detailed information about a selected piece of Mars real estate.  \* It sets this information in the [DetailViewModel], which it gets as a Parcelable property  \* through Jetpack Navigation's SafeArgs.  \*/* class DetailFragment : Fragment() {  override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  savedInstanceState: Bundle?): View? {  val application = *requireNotNull*(*activity*).*application* val binding = FragmentDetailBinding.inflate(inflater)  binding.*lifecycleOwner* = this  val marsProperty = DetailFragmentArgs.fromBundle(*arguments*!!).*selectedProperty* val viewModelFactory = DetailViewModelFactory(marsProperty, application)  binding.*viewModel* = ViewModelProvider(  this, viewModelFactory).get(DetailViewModel::class.*java*)  return binding.*root* } } |

Tabel 11. Source Code Soal 1 Kotlin yang pertama

**DetailViewModel.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*/  package com.example.android.marsrealestate.detail  import android.app.Application import androidx.lifecycle.AndroidViewModel import androidx.lifecycle.LiveData import androidx.lifecycle.MutableLiveData import androidx.lifecycle.Transformations import androidx.lifecycle.ViewModel import com.example.android.marsrealestate.R import com.example.android.marsrealestate.network.MarsProperty  */\*\*  \* The [ViewModel] associated with the [DetailFragment], containing information about the selected  \* [MarsProperty].  \*/* class DetailViewModel(marsProperty: MarsProperty, app: Application) : AndroidViewModel(app) {  private val \_selectedProperty = MutableLiveData<MarsProperty>()   // The external LiveData for the SelectedProperty  val selectedProperty: LiveData<MarsProperty>  get() = \_selectedProperty   // Initialize the \_selectedProperty MutableLiveData  init {  \_selectedProperty.*value* = marsProperty  }   // The displayPropertyPrice formatted Transformation Map LiveData, which displays the sale  // or rental price.  val displayPropertyPrice = Transformations.map(selectedProperty) **{** app.*applicationContext*.getString(  when (**it**.isRental) {  true -> R.string.*display\_price\_monthly\_rental* false -> R.string.*display\_price* }, **it**.price)  **}** // The displayPropertyType formatted Transformation Map LiveData, which displays the  // "For Rent/Sale" String  val displayPropertyType = Transformations.map(selectedProperty) **{** app.*applicationContext*.getString(R.string.*display\_type*,  app.*applicationContext*.getString(  when(**it**.isRental) {  true -> R.string.*type\_rent* false -> R.string.*type\_sale* }))  **}** } |

Tabel 12. Source Code Soal 1 Kotlin yang kedua

**DetailViewModelFactory.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*/  package com.example.android.marsrealestate.detail  import android.app.Application import androidx.lifecycle.ViewModel import androidx.lifecycle.ViewModelProvider import com.example.android.marsrealestate.network.MarsProperty  */\*\*  \* Simple ViewModel factory that provides the MarsProperty and context to the ViewModel.  \*/* class DetailViewModelFactory(  private val marsProperty: MarsProperty,  private val application: Application) : ViewModelProvider.Factory {  @Suppress("unchecked\_cast")  override fun <T : ViewModel> create(modelClass: Class<T>): T {  if (modelClass.isAssignableFrom(DetailViewModel::class.*java*)) {  return DetailViewModel(marsProperty, application) as T  }  throw IllegalArgumentException("Unknown ViewModel class")  } } |

Tabel 13. Source Code Soal 1 Kotlin yang ketiga

**MarsApiService.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*  \*/  package com.example.android.marsrealestate.network  // import com.jakewharton.retrofit2.adapter.kotlin.coroutines.CoroutineCallAdapterFactory import com.squareup.moshi.Moshi import com.squareup.moshi.kotlin.reflect.KotlinJsonAdapterFactory // import kotlinx.coroutines.Deferred import retrofit2.Retrofit import retrofit2.converter.moshi.MoshiConverterFactory import retrofit2.http.GET import retrofit2.http.Query  private const val *BASE\_URL* = "https://mars.udacity.com/" enum class MarsApiFilter(val value: String) { *SHOW\_RENT*("rent"), *SHOW\_BUY*("buy"), *SHOW\_ALL*("all") }  */\*\*  \* Build the Moshi object that Retrofit will be using, making sure to add the Kotlin adapter for  \* full Kotlin compatibility.  \*/* private val *moshi* = Moshi.Builder()  .add(KotlinJsonAdapterFactory())  .build()  */\*\*  \* Use the Retrofit builder to build a retrofit object using a Moshi converter with our Moshi  \* object.  \*/* private val *retrofit* = Retrofit.Builder()  .addConverterFactory(MoshiConverterFactory.create(*moshi*))  .baseUrl(*BASE\_URL*)  .build()  */\*\*  \* A public interface that exposes the [getProperties] method  \*/* interface MarsApiService {  */\*\*  \* Returns a Coroutine [List] of [MarsProperty] which can be fetched with await() if in a Coroutine scope.  \* The @GET annotation indicates that the "realestate" endpoint will be requested with the GET  \* HTTP method  \*/* @GET("realestate")  suspend fun getProperties(@Query("filter") type: String): List<MarsProperty> }  */\*\*  \* A public Api object that exposes the lazy-initialized Retrofit service  \*/* object MarsApi {  val retrofitService : MarsApiService by *lazy* **{** *retrofit*.create(MarsApiService::class.*java*) **}** } |

Tabel 14. Source Code Soal 1 Kotlin yang keempat

**MarsProperty.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*  \*/  package com.example.android.marsrealestate.network  import android.os.Parcelable import androidx.lifecycle.LiveData import com.example.android.marsrealestate.overview.MarsApiStatus import com.squareup.moshi.Json import kotlinx.android.parcel.Parcelize  */\*\*  \* Gets Mars real estate property information from the Mars API Retrofit service and updates the  \* [MarsProperty] and [MarsApiStatus] [LiveData]. The Retrofit service returns a coroutine  \* Deferred, which we await to get the result of the transaction.  \* @param filter the [MarsApiFilter] that is sent as part of the web server request  \*/* @Parcelize data class MarsProperty(  val id: String,  // used to map img\_src from the JSON to imgSrcUrl in our class  @Json(name = "img\_src") val imgSrcUrl: String,  val type: String,  val price: Double) : Parcelable {  val isRental  get() = type == "rent" } |

Tabel 15. Source Code Soal 1 Kotlin yang kelima

**BindingAdapters.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*  \*/  package com.example.android.marsrealestate  import android.view.View import android.widget.ImageView import android.widget.TextView import androidx.core.net.toUri import androidx.databinding.BindingAdapter import androidx.recyclerview.widget.RecyclerView import com.bumptech.glide.Glide import com.bumptech.glide.request.RequestOptions import com.example.android.marsrealestate.network.MarsProperty import com.example.android.marsrealestate.overview.MarsApiStatus import com.example.android.marsrealestate.overview.PhotoGridAdapter  */\*\*  \* When there is no Mars property data (data is null), hide the [RecyclerView], otherwise show it.  \*/* @BindingAdapter("listData") fun bindRecyclerView(recyclerView: RecyclerView, data: List<MarsProperty>?) {  val adapter = recyclerView.*adapter* as PhotoGridAdapter  adapter.submitList(data) }  */\*\*  \* Uses the Glide library to load an image by URL into an [ImageView]  \*/* @BindingAdapter("imageUrl") fun bindImage(imgView: ImageView, imgUrl: String?) {  imgUrl?.*let* **{** val imgUri = imgUrl.*toUri*().buildUpon().scheme("https").build()  Glide.with(imgView.*context*)  .load(imgUri)  .apply(RequestOptions()  .placeholder(R.drawable.*loading\_animation*)  .error(R.drawable.*ic\_broken\_image*))  .into(imgView)  **}** }  */\*\*  \* This binding adapter displays the [MarsApiStatus] of the network request in an image view. When  \* the request is loading, it displays a loading\_animation. If the request has an error, it  \* displays a broken image to reflect the connection error. When the request is finished, it  \* hides the image view.  \*/* @BindingAdapter("marsApiStatus") fun bindStatus(statusImageView: ImageView, status: MarsApiStatus?) {  when (status) {  MarsApiStatus.*LOADING* -> {  statusImageView.*visibility* = View.*VISIBLE* statusImageView.setImageResource(R.drawable.*loading\_animation*)  }  MarsApiStatus.*ERROR* -> {  statusImageView.*visibility* = View.*VISIBLE* statusImageView.setImageResource(R.drawable.*ic\_connection\_error*)  }  MarsApiStatus.*DONE* -> {  statusImageView.*visibility* = View.*GONE* }  } } |

Tabel 16. Source Code Soal 1 Kotlin yang keenam

**MainActivity.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | /\*  \* Copyright 2018, The Android Open Source Project  \*  \* Licensed under the Apache License, Version 2.0 (the "License");  \* you may not use this file except in compliance with the License.  \* You may obtain a copy of the License at  \*  \* http://www.apache.org/licenses/LICENSE-2.0  \*  \* Unless required by applicable law or agreed to in writing, software  \* distributed under the License is distributed on an "AS IS" BASIS,  \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  \* See the License for the specific language governing permissions and  \* limitations under the License.  \*  \*/  package com.example.android.marsrealestate  import android.os.Bundle import androidx.appcompat.app.AppCompatActivity  class MainActivity : AppCompatActivity() {   */\*\*  \* Our MainActivity is only responsible for setting the content view that contains the  \* Navigation Host.  \*/* override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*activity\_main*)  } } |

Tabel 17. Source Code Soal 1 Kotlin yang ketujuh

**activity\_main.xml**

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

Tabel 18. Source Code Soal 1 XML yang pertama

**fragment\_detail.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77 | <?xml version="1.0" encoding="utf-8"?>  <!--  ~ Copyright 2018, The Android Open Source Project  ~  ~ Licensed under the Apache License, Version 2.0 (the "License");  ~ you may not use this file except in compliance with the License.  ~ You may obtain a copy of the License at  ~  ~ http://www.apache.org/licenses/LICENSE-2.0  ~  ~ Unless required by applicable law or agreed to in writing, software  ~ distributed under the License is distributed on an "AS IS" BASIS,  ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  ~ See the License for the specific language governing permissions and  ~ limitations under the License.  ~  -->  <layout 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">   <data>  <variable  name="viewModel"  type="com.example.android.marsrealestate.detail.DetailViewModel" />  </data>   <ScrollView  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".DetailFragment">   <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:padding="16dp">   <ImageView  android:id="@+id/main\_photo\_image"  android:layout\_width="0dp"  android:layout\_height="266dp"  android:scaleType="centerCrop"  app:imageUrl="@{viewModel.selectedProperty.imgSrcUrl}"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  tools:src="@tools:sample/backgrounds/scenic" />   <TextView  android:id="@+id/property\_type\_text"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="16dp"  android:text="@{viewModel.displayPropertyType}"  android:textColor="#de000000"  android:textSize="39sp"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/main\_photo\_image"  tools:text="To Rent" />   <TextView  android:id="@+id/price\_value\_text"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="8dp"  android:text="@{viewModel.displayPropertyPrice}"  android:textColor="#de000000"  android:textSize="20sp"  app:layout\_constraintStart\_toStartOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/property\_type\_text"  tools:text="$100,000" />   </androidx.constraintlayout.widget.ConstraintLayout>  </ScrollView> </layout> |

Tabel 19. Source Code Soal 1 XML yang kedua

**fragment\_overview.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | <?xml version="1.0" encoding="utf-8"?>  <!--  ~ Copyright 2018, The Android Open Source Project  ~  ~ Licensed under the Apache License, Version 2.0 (the "License");  ~ you may not use this file except in compliance with the License.  ~ You may obtain a copy of the License at  ~  ~ http://www.apache.org/licenses/LICENSE-2.0  ~  ~ Unless required by applicable law or agreed to in writing, software  ~ distributed under the License is distributed on an "AS IS" BASIS,  ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  ~ See the License for the specific language governing permissions and  ~ limitations under the License.  ~  -->  <layout 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">   <data>  <variable  name="viewModel"  type="com.example.android.marsrealestate.overview.OverviewViewModel" />  </data>   <androidx.constraintlayout.widget.ConstraintLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="com.example.android.marsrealestate.MainActivity">   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/photos\_grid"  android:layout\_width="0dp"  android:layout\_height="0dp"  android:padding="6dp"  android:clipToPadding="false"  app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:listData="@{viewModel.properties}"  app:spanCount="2"  tools:itemCount="16"  tools:listitem="@layout/grid\_view\_item" />   <ImageView  android:id="@+id/status\_image"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toTopOf="parent"  app:marsApiStatus="@{viewModel.status}" />   </androidx.constraintlayout.widget.ConstraintLayout> </layout> |

Tabel 20. Source Code Soal 1 XML yang ketiga

**grid\_view\_item.xml**

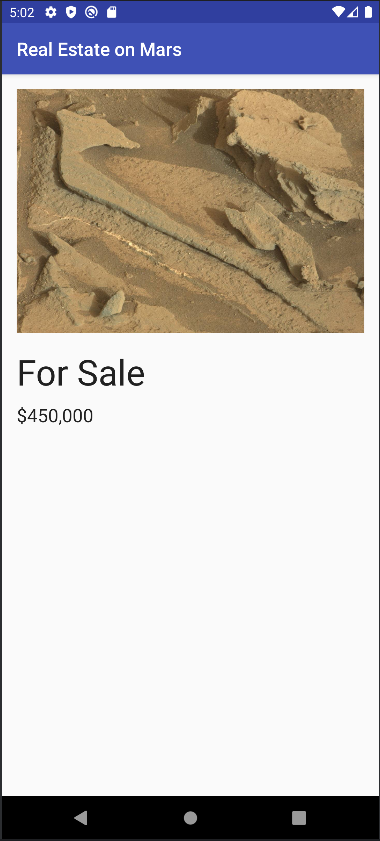
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37 | <?xml version="1.0" encoding="utf-8"?>  <!--  ~ Copyright 2018, The Android Open Source Project  ~  ~ Licensed under the Apache License, Version 2.0 (the "License");  ~ you may not use this file except in compliance with the License.  ~ You may obtain a copy of the License at  ~  ~ http://www.apache.org/licenses/LICENSE-2.0  ~  ~ Unless required by applicable law or agreed to in writing, software  ~ distributed under the License is distributed on an "AS IS" BASIS,  ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  ~ See the License for the specific language governing permissions and  ~ limitations under the License.  ~  -->  <layout 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">  <data>  <variable  name="property"  type="com.example.android.marsrealestate.network.MarsProperty" />  </data>  <ImageView  android:id="@+id/mars\_image"  android:layout\_width="match\_parent"  android:layout\_height="170dp"  android:scaleType="centerCrop"  android:adjustViewBounds="true"  android:padding="2dp"  app:imageUrl="@{property.imgSrcUrl}"  tools:src="@tools:sample/backgrounds/scenic"/> </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.marsrealestate.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 (imgUrl) dan memuatnya ke dalam ImageView (imgView) menggunakan Glide library. Jika imgUrl 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 AppCompat 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 AppCompat 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 konstraints.

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 AppCompat 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:listData 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 AppCompat 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/Modul%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  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package com.example.praktikum.frly.kotlin.shrine.application  import android.app.Application import androidx.appcompat.app.AppCompatDelegate  class ShrineApplication : Application() {  companion object {  lateinit var instance: ShrineApplication  private set  }   override fun onCreate() {  super.onCreate()  instance = this   AppCompatDelegate.setCompatVectorFromResourcesEnabled(true)  }  } |

Tabel 22. Source Code Soal 1 Kotlin yang pertama

**ImageRequester.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | package com.example.praktikum.frly.kotlin.shrine.network  import android.content.Context import android.graphics.Bitmap import android.util.LruCache  import com.android.volley.RequestQueue import com.android.volley.toolbox.ImageLoader import com.android.volley.toolbox.NetworkImageView import com.android.volley.toolbox.Volley import com.example.praktikum.frly.kotlin.shrine.application.ShrineApplication  */\*\*  \* Class that handles image requests using Volley.  \*/* object ImageRequester {  private val requestQueue: RequestQueue  private val imageLoader: ImageLoader  private val maxByteSize: Int   init {  val context = ShrineApplication.instance  requestQueue = Volley.newRequestQueue(context)  requestQueue.start()  maxByteSize = calculateMaxByteSize(context)  imageLoader = ImageLoader(  requestQueue,  object : ImageLoader.ImageCache {  private val lruCache = object : LruCache<String, Bitmap>(maxByteSize) {  override fun sizeOf(url: String, bitmap: Bitmap): Int {  return bitmap.*byteCount* }  }   @Synchronized  override fun getBitmap(url: String): Bitmap? {  return lruCache.get(url)  }   @Synchronized  override fun putBitmap(url: String, bitmap: Bitmap) {  lruCache.put(url, bitmap)  }  })  }   */\*\*  \* Sets the image on the given [NetworkImageView] to the image at the given URL  \*  \* @param networkImageView [NetworkImageView] to set image on  \* @param url URL of the image  \*/* fun setImageFromUrl(networkImageView: NetworkImageView, url: String) {  networkImageView.setImageUrl(url, imageLoader)  }   private fun calculateMaxByteSize(context: Context): Int {  val displayMetrics = context.*resources*.*displayMetrics* val screenBytes = displayMetrics.widthPixels \* displayMetrics.heightPixels \* 4  return screenBytes \* 3  } } |

Tabel 23. Source Code Soal 1 Kotlin yang kedua

**ProductEntry.kt**

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

Tabel 24. Source Code Soal 1 Kotlin yang ketiga

**StaggeredProductCardRecyclerViewAdapter.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | package com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout  import android.view.LayoutInflater import android.view.ViewGroup import androidx.recyclerview.widget.RecyclerView  import com.example.praktikum.frly.kotlin.shrine.R import com.example.praktikum.frly.kotlin.shrine.network.ImageRequester import com.example.praktikum.frly.kotlin.shrine.network.ProductEntry  */\*\*  \* Adapter used to show an asymmetric grid of products, with 2 items in the first column, and 1  \* item in the second column, and so on.  \*/* class StaggeredProductCardRecyclerViewAdapter(private val productList: List<ProductEntry>?) : RecyclerView.Adapter<StaggeredProductCardViewHolder>() {   override fun getItemViewType(position: Int): Int {  return position % 3  }   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): StaggeredProductCardViewHolder {  var layoutId = R.layout.*shr\_staggered\_product\_card\_first* if (viewType == 1) {  layoutId = R.layout.*shr\_staggered\_product\_card\_second* } else if (viewType == 2) {  layoutId = R.layout.*shr\_staggered\_product\_card\_third* }   val layoutView = LayoutInflater.from(parent.*context*).inflate(layoutId, parent, false)  return StaggeredProductCardViewHolder(layoutView)  }   override fun onBindViewHolder(holder: StaggeredProductCardViewHolder, position: Int) {  if (productList != null && position < productList.size) {  val product = productList[position]  holder.productTitle.*text* = product.title  holder.productPrice.*text* = product.price  ImageRequester.setImageFromUrl(holder.productImage, product.url)  }  }   override fun getItemCount(): Int {  return productList?.size ?: 0  } } |

Tabel 25. Source Code Soal 1 Kotlin yang keempat

**StaggeredProductCardViewHolder.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout  import android.view.View import android.widget.TextView import androidx.recyclerview.widget.RecyclerView  import com.android.volley.toolbox.NetworkImageView import com.example.praktikum.frly.kotlin.shrine.R  class StaggeredProductCardViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {   var productImage: NetworkImageView = itemView.findViewById(R.id.*product\_image*)  var productTitle: TextView = itemView.findViewById(R.id.*product\_title*)  var productPrice: TextView = itemView.findViewById(R.id.*product\_price*) } |

Tabel 26. Source Code Soal 1 Kotlin yang kelima

**LoginFragment.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51 | package com.example.praktikum.frly.kotlin.shrine  import android.os.Bundle import android.text.Editable import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import android.widget.Button import android.widget.EditText import androidx.fragment.app.Fragment import com.google.android.material.textfield.TextInputLayout import com.example.praktikum.frly.kotlin.shrine.R  */\*\*  \* Fragment representing the login screen for Shrine.  \*/* class LoginFragment : Fragment() {   override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  // Inflate the layout for this fragment  val view = inflater.inflate(R.layout.*shr\_login\_fragment*, container, false)   // Set an error if the password is less than 8 characters.  view.findViewById<Button>(R.id.*next\_button*).setOnClickListener **{** if (!isPasswordValid(view.findViewById<EditText>(R.id.*password\_edit\_text*).*text*)) {  view.findViewById<TextInputLayout>(R.id.*password\_text\_input*).*error* = getString(R.string.*shr\_error\_password*)  } else {  view.findViewById<TextInputLayout>(R.id.*password\_text\_input*).*error* = null // Clear the error  (*activity* as NavigationHost).navigateTo(ProductGridFragment(), false) // Navigate to the next Fragment  }  **}** // Clear the error once more than 8 characters are typed.  view.findViewById<EditText>(R.id.*password\_edit\_text*).setOnKeyListener **{** \_, \_, \_ **->** if (isPasswordValid(view.findViewById<EditText>(R.id.*password\_edit\_text*).*text*)) {  view.findViewById<TextInputLayout>(R.id.*password\_text\_input*).*error* = null //Clear the error  }  false  **}** return view  }   /\*  In reality, this will have more complex logic including, but not limited to, actual  authentication of the username and password.  \*/  private fun isPasswordValid(text: Editable?): Boolean {  return text != null && text.length >= 8  } } |

Tabel 27. Source Code Soal 1 Kotlin yang keenam

**MainActivity.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | package com.example.praktikum.frly.kotlin.shrine  import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.fragment.app.Fragment import com.example.praktikum.frly.kotlin.shrine.R  class MainActivity : AppCompatActivity(), NavigationHost {   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setContentView(R.layout.*shr\_main\_activity*)   if (savedInstanceState == null) {  *supportFragmentManager* .beginTransaction()  .add(R.id.*container*, LoginFragment())  .commit()  }  }   */\*\*  \* Navigate to the given fragment.  \*  \* @param fragment Fragment to navigate to.  \* @param addToBackstack Whether or not the current fragment should be added to the backstack.  \*/* override fun navigateTo(fragment: Fragment, addToBackstack: Boolean) {  val transaction = *supportFragmentManager* .beginTransaction()  .replace(R.id.*container*, fragment)   if (addToBackstack) {  transaction.addToBackStack(null)  }   transaction.commit()  } } |

Tabel 28. Source Code Soal 1 Kotlin yang ketujuh

**NavigationHost.kt**

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

Tabel 29. Source Code Soal 1 Kotlin yang kedelapan

**NavigationIconClickListener.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  40  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65 | package com.example.praktikum.frly.kotlin.shrine  import android.animation.AnimatorSet import android.animation.ObjectAnimator import android.app.Activity import android.content.Context import android.graphics.drawable.Drawable import android.util.DisplayMetrics import android.view.View import android.view.animation.Interpolator import android.widget.ImageView import com.example.praktikum.frly.kotlin.shrine.R  */\*\*  \* [android.view.View.OnClickListener] used to translate the product grid sheet downward on  \* the Y-axis when the navigation icon in the toolbar is pressed.  \*/* class NavigationIconClickListener @JvmOverloads internal constructor(  private val context: Context, private val sheet: View, private val interpolator: Interpolator? = null,  private val openIcon: Drawable? = null, private val closeIcon: Drawable? = null) : View.OnClickListener {   private val animatorSet = AnimatorSet()  private val height: Int  private var backdropShown = false   init {  val displayMetrics = DisplayMetrics()  (context as Activity).*windowManager*.*defaultDisplay*.getMetrics(displayMetrics)  height = displayMetrics.heightPixels  }   override fun onClick(view: View) {  backdropShown = !backdropShown   // Cancel the existing animations  animatorSet.removeAllListeners()  animatorSet.end()  animatorSet.cancel()   updateIcon(view)   val translateY = height - context.*resources*.getDimensionPixelSize(R.dimen.*shr\_product\_grid\_reveal\_height*)   val animator = ObjectAnimator.ofFloat(sheet, "translationY", (if (backdropShown) translateY else 0).toFloat())  animator.*duration* = 500  if (interpolator != null) {  animator.*interpolator* = interpolator  }  animatorSet.play(animator)  animator.start()  }   private fun updateIcon(view: View) {  if (openIcon != null && closeIcon != null) {  if (view !is ImageView) {  throw IllegalArgumentException("updateIcon() must be called on an ImageView")  }  if (backdropShown) {  view.setImageDrawable(closeIcon)  } else {  view.setImageDrawable(openIcon)  }  }  } } |

Tabel 30. Source Code Soal 1 Kotlin yang kesembilan

**ProductCardRecyclerViewAdapter.kt**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | package com.example.praktikum.frly.kotlin.shrine  import android.view.LayoutInflater import android.view.ViewGroup import androidx.recyclerview.widget.RecyclerView import com.example.praktikum.frly.kotlin.shrine.R import com.example.praktikum.frly.kotlin.shrine.network.ImageRequester  import com.example.praktikum.frly.kotlin.shrine.network.ProductEntry  */\*\*  \* Adapter used to show a simple grid of products.  \*/* class ProductCardRecyclerViewAdapter internal constructor(private val productList: List<ProductEntry>) : RecyclerView.Adapter<ProductCardViewHolder>() {   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProductCardViewHolder {  val layoutView = LayoutInflater.from(parent.*context*).inflate(R.layout.*shr\_product\_card*, parent, false)  return ProductCardViewHolder(layoutView)  }   override fun onBindViewHolder(holder: ProductCardViewHolder, position: Int) {  if (position < productList.size) {  val product = productList[position]  holder.productTitle.*text* = product.title  holder.productPrice.*text* = product.price  ImageRequester.setImageFromUrl(holder.productImage, product.url)  }  }   override fun getItemCount(): Int {  return productList.size  } } |

Tabel 31. Source Code Soal 1 Kotlin yang kesepuluh

**ProductCardViewHolder.kt**

|  |  |
| --- | --- |
| 1  2  3  5  6  7  8  9  10  11  12  13  14  15 | package com.example.praktikum.frly.kotlin.shrine  import android.view.View import android.widget.TextView import androidx.recyclerview.widget.RecyclerView  import com.android.volley.toolbox.NetworkImageView import com.example.praktikum.frly.kotlin.shrine.R  class ProductCardViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {   var productImage: NetworkImageView = itemView.findViewById(R.id.*product\_image*)  var productTitle: TextView = itemView.findViewById(R.id.*product\_title*)  var productPrice: TextView = itemView.findViewById(R.id.*product\_price*) } |

Tabel 32. Source Code Soal 1 Kotlin yang kesebelas

**ProductGridFragment.kt**

|  |  |
| --- | --- |
| 1  2  3  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74 | package com.example.praktikum.frly.kotlin.shrine  import android.os.Bundle import android.view.LayoutInflater import android.view.Menu import android.view.MenuInflater import android.view.View import android.view.ViewGroup import android.view.animation.AccelerateDecelerateInterpolator import androidx.appcompat.app.AppCompatActivity import androidx.appcompat.widget.Toolbar import androidx.core.content.ContextCompat import androidx.core.widget.NestedScrollView import androidx.fragment.app.Fragment import androidx.recyclerview.widget.GridLayoutManager import androidx.recyclerview.widget.RecyclerView import com.example.praktikum.frly.kotlin.shrine.R import com.example.praktikum.frly.kotlin.shrine.network.ProductEntry import com.example.praktikum.frly.kotlin.shrine.staggeredgridlayout.StaggeredProductCardRecyclerViewAdapter  class ProductGridFragment : Fragment() {   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  setHasOptionsMenu(true)  }   override fun onCreateView(  inflater: LayoutInflater, container: ViewGroup?, savedInstanceState: Bundle?): View? {  // Inflate the layout for this fragment with the ProductGrid theme  val view = inflater.inflate(R.layout.*shr\_product\_grid\_fragment*, container, false)   val app\_bar = view.findViewById<Toolbar>(R.id.*app\_bar*)  val recycler\_view = view.findViewById<RecyclerView>(R.id.*recycler\_view*)  val product\_grid = view.findViewById<NestedScrollView>(R.id.*product\_grid*)   // Set up the tool bar  (*activity* as AppCompatActivity).setSupportActionBar(app\_bar)  app\_bar.setNavigationOnClickListener(  NavigationIconClickListener(  requireActivity(),  product\_grid,  AccelerateDecelerateInterpolator(),  ContextCompat.getDrawable(requireContext(), R.drawable.*shr\_branded\_menu*), // Menu open icon  ContextCompat.getDrawable(requireContext(), R.drawable.*shr\_close\_menu*))  ) // Menu close icon   // Set up the RecyclerView  recycler\_view.setHasFixedSize(true)  val gridLayoutManager = GridLayoutManager(*context*, 2, RecyclerView.*HORIZONTAL*, false)  gridLayoutManager.*spanSizeLookup* = object : GridLayoutManager.SpanSizeLookup() {  override fun getSpanSize(position: Int): Int {  return if (position % 3 == 2) 2 else 1  }  }  recycler\_view.*layoutManager* = gridLayoutManager  val adapter = StaggeredProductCardRecyclerViewAdapter(  ProductEntry.initProductEntryList(*resources*))  recycler\_view.*adapter* = adapter  val largePadding = *resources*.getDimensionPixelSize(R.dimen.*shr\_staggered\_product\_grid\_spacing\_large*)  val smallPadding = *resources*.getDimensionPixelSize(R.dimen.*shr\_staggered\_product\_grid\_spacing\_small*)  recycler\_view.addItemDecoration(ProductGridItemDecoration(largePadding, smallPadding))   // Set cut corner background for API 23+  product\_grid.*background* = *context*?.getDrawable(R.drawable.*shr\_product\_grid\_background\_shape*)   return view  }   override fun onCreateOptionsMenu(menu: Menu, menuInflater: MenuInflater) {  menuInflater.inflate(R.menu.*shr\_toolbar\_menu*, menu)  super.onCreateOptionsMenu(menu, menuInflater)  } } |

Tabel 33. Source Code Soal 1 Kotlin yang kedua belas

**ProductGridItemDecoration.kt**

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

Tabel 34. Source Code Soal 1 Kotlin yang ketiga belas

**shr\_backdrop.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | <?xml version="1.0" encoding="utf-8"?> <merge xmlns:android="http://schemas.android.com/apk/res/android">   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_featured\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_apartment\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_accessories\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_shoes\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_tops\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_bottoms\_label" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_dresses\_label" />   <View  android:layout\_width="56dp"  android:layout\_height="1dp"  android:layout\_margin="16dp"  android:background="?android:attr/textColorPrimary" />   <com.google.android.material.button.MaterialButton  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="@string/shr\_account\_label" />  </merge> |

Tabel 35. Source Code Soal 1 XML yang pertama

**shr\_login\_fragment.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91 | <?xml version="1.0" encoding="utf-8"?> <ScrollView 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"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:background="@color/loginPageBackgroundColor"  tools:context="com.example.praktikum.frly.kotlin.shrine.LoginFragment">   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:clipChildren="false"  android:clipToPadding="false"  android:orientation="vertical"  android:padding="24dp"  android:paddingTop="16dp">   <ImageView  android:layout\_width="64dp"  android:layout\_height="64dp"  android:layout\_gravity="center\_horizontal"  android:layout\_marginTop="48dp"  android:layout\_marginBottom="16dp"  app:srcCompat="@drawable/shr\_logo"  android:contentDescription="@string/shr\_logo\_content\_description"  app:tint="?android:attr/textColorPrimary" />   <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_gravity="center\_horizontal"  android:layout\_marginBottom="132dp"  android:text="@string/shr\_app\_name"  android:textAppearance="@style/TextAppearance.Shrine.Title" />   <com.google.android.material.textfield.TextInputLayout  style="@style/Widget.Shrine.TextInputLayout"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/shr\_hint\_username">   <com.google.android.material.textfield.TextInputEditText  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:inputType="text"  android:maxLines="1" />  </com.google.android.material.textfield.TextInputLayout>   <com.google.android.material.textfield.TextInputLayout  android:id="@+id/password\_text\_input"  style="@style/Widget.Shrine.TextInputLayout"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/shr\_hint\_password"  app:errorEnabled="true">   <com.google.android.material.textfield.TextInputEditText  android:id="@+id/password\_edit\_text"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:inputType="textPassword" />  </com.google.android.material.textfield.TextInputLayout>   <RelativeLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content">   <com.google.android.material.button.MaterialButton  android:id="@+id/next\_button"  style="@style/Widget.Shrine.Button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_alignParentEnd="true"  android:layout\_alignParentRight="true"  android:text="@string/shr\_button\_next" />   <com.google.android.material.button.MaterialButton  android:id="@+id/cancel\_button"  style="@style/Widget.Shrine.Button.TextButton"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginEnd="12dp"  android:layout\_marginRight="12dp"  android:layout\_toStartOf="@id/next\_button"  android:layout\_toLeftOf="@id/next\_button"  android:text="@string/shr\_button\_cancel" />   </RelativeLayout>  </LinearLayout> </ScrollView> |

Tabel 36. Source Code Soal 1 XML yang kedua

**shr\_main\_activity.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?xml version="1.0" encoding="utf-8"?> <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:id="@+id/container"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="com.example.praktikum.frly.kotlin.shrine.MainActivity" /> |

Tabel 37. Source Code Soal 1 XML yang ketiga

**shr\_product\_card.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45 | <?xml version="1.0" encoding="utf-8"?> <com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  app:cardBackgroundColor="@android:color/transparent"  app:cardElevation="0dp"  app:cardPreventCornerOverlap="true">   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical">   <com.android.volley.toolbox.NetworkImageView  android:id="@+id/product\_image"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:background="?attr/colorPrimaryDark"  android:scaleType="centerCrop" />   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:padding="16dp">   <TextView  android:id="@+id/product\_title"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_title"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceSubtitle2" />   <TextView  android:id="@+id/product\_price"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_description"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceBody2" />  </LinearLayout>  </LinearLayout> </com.google.android.material.card.MaterialCardView> |

Tabel 38. Source Code Soal 1 XML yang keempat

**shr\_product\_grid\_fragment.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  40  50  51  52  53  54  55  56  57 | <?xml version="1.0" encoding="utf-8"?> <FrameLayout 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"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:clipChildren="false"  android:clipToPadding="false"  tools:context="com.example.praktikum.frly.kotlin.shrine.ProductGridFragment">   <LinearLayout  style="@style/Widget.Shrine.Backdrop"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:gravity="center\_horizontal"  android:orientation="vertical"  android:paddingTop="88dp">   <include layout="@layout/shr\_backdrop" />  </LinearLayout>   <com.google.android.material.appbar.AppBarLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  app:elevation="0dp">   <androidx.appcompat.widget.Toolbar  android:id="@+id/app\_bar"  style="@style/Widget.Shrine.Toolbar"  android:layout\_width="match\_parent"  android:layout\_height="?attr/actionBarSize"  android:paddingStart="12dp"  android:paddingLeft="12dp"  android:paddingEnd="12dp"  android:paddingRight="12dp"  app:contentInsetStart="0dp"  app:navigationIcon="@drawable/shr\_branded\_menu"  app:title="@string/shr\_app\_name" />  </com.google.android.material.appbar.AppBarLayout>   <androidx.core.widget.NestedScrollView  android:id="@+id/product\_grid"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_marginTop="56dp"  android:background="@color/productGridBackgroundColor"  android:elevation="8dp"  app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior">   <androidx.recyclerview.widget.RecyclerView  android:id="@+id/recycler\_view"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />   </androidx.core.widget.NestedScrollView>  </FrameLayout> |

Tabel 39. Source Code Soal 1 XML yang kelima

**shr\_staggered\_product\_card\_first.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | <?xml version="1.0" encoding="utf-8"?> <com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  app:cardBackgroundColor="@android:color/transparent"  app:cardElevation="0dp"  app:cardPreventCornerOverlap="true"  app:contentPaddingTop="@dimen/shr\_staggered\_product\_grid\_margin\_top\_first">   <!-- Product card layout used in MDC-103 and onward -->  <LinearLayout  android:layout\_width="@dimen/shr\_staggered\_product\_grid\_card\_width\_landscape"  android:layout\_height="wrap\_content"  android:layout\_marginStart="@dimen/shr\_staggered\_product\_grid\_spacing\_small"  android:layout\_marginLeft="@dimen/shr\_staggered\_product\_grid\_spacing\_small"  android:orientation="vertical">   <com.android.volley.toolbox.NetworkImageView  android:id="@+id/product\_image"  android:layout\_width="match\_parent"  android:layout\_height="@dimen/shr\_staggered\_product\_grid\_card\_height\_landscape"  android:background="?attr/colorPrimaryDark"  android:scaleType="centerCrop" />   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:padding="16dp">   <TextView  android:id="@+id/product\_title"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_title"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceSubtitle2" />   <TextView  android:id="@+id/product\_price"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_description"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceBody2" />  </LinearLayout>  </LinearLayout> </com.google.android.material.card.MaterialCardView> |

Tabel 40. Source Code Soal 1 XML yang keenam

**shr\_staggered\_product\_card\_second.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49 | <?xml version="1.0" encoding="utf-8"?> <com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  app:cardBackgroundColor="@android:color/transparent"  app:cardElevation="0dp"  app:cardPreventCornerOverlap="true"  app:contentPaddingTop="@dimen/shr\_staggered\_product\_grid\_margin\_top\_second">   <!-- Product card layout used in MDC-103 and onward -->  <LinearLayout  android:layout\_width="@dimen/shr\_staggered\_product\_grid\_card\_width\_landscape"  android:layout\_height="wrap\_content"  android:layout\_marginEnd="@dimen/shr\_staggered\_product\_grid\_spacing\_small"  android:layout\_marginRight="@dimen/shr\_staggered\_product\_grid\_spacing\_small"  android:orientation="vertical">   <com.android.volley.toolbox.NetworkImageView  android:id="@+id/product\_image"  android:layout\_width="match\_parent"  android:layout\_height="@dimen/shr\_staggered\_product\_grid\_card\_height\_landscape"  android:background="?attr/colorPrimaryDark"  android:scaleType="centerCrop" />   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:padding="16dp">   <TextView  android:id="@+id/product\_title"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_title"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceSubtitle2" />   <TextView  android:id="@+id/product\_price"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_description"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceBody2" />  </LinearLayout>  </LinearLayout> </com.google.android.material.card.MaterialCardView> |

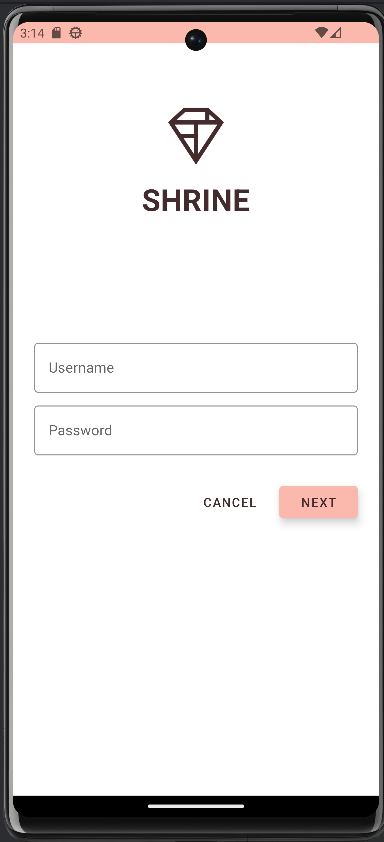
Tabel 41. Source Code Soal 1 XML yang ketujuh

**shr\_staggered\_product\_card\_third.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | <?xml version="1.0" encoding="utf-8"?> <com.google.android.material.card.MaterialCardView xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:layout\_width="@dimen/shr\_staggered\_product\_grid\_card\_width\_portrait"  android:layout\_height="match\_parent"  app:cardBackgroundColor="@android:color/transparent"  app:cardElevation="0dp"  app:cardPreventCornerOverlap="true"  app:contentPaddingTop="@dimen/shr\_staggered\_product\_grid\_margin\_top\_third">   <!-- Product card layout used in MDC-103 and onward -->  <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical">   <com.android.volley.toolbox.NetworkImageView  android:id="@+id/product\_image"  android:layout\_width="match\_parent"  android:layout\_height="@dimen/shr\_staggered\_product\_grid\_card\_height\_portrait"  android:background="?attr/colorPrimaryDark"  android:scaleType="centerCrop" />   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:padding="16dp">   <TextView  android:id="@+id/product\_title"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_title"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceSubtitle2" />   <TextView  android:id="@+id/product\_price"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/shr\_product\_description"  android:textAlignment="center"  android:textAppearance="?attr/textAppearanceBody2" />  </LinearLayout>  </LinearLayout> </com.google.android.material.card.MaterialCardView> |

Tabel 42. Source Code Soal 1 XML yang kedelepan

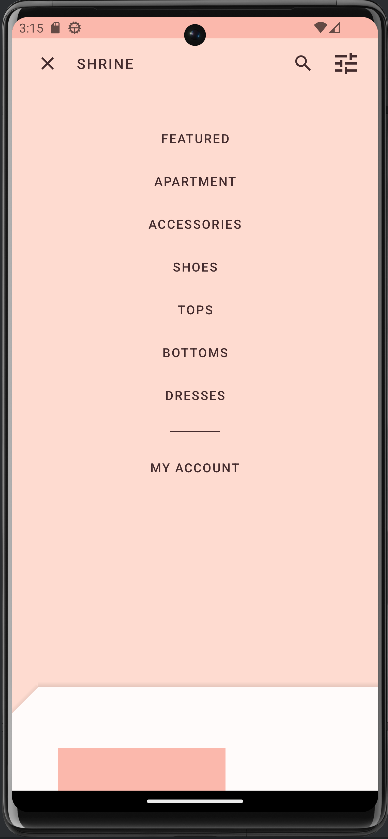
### B. Output Program



Gambar 23. Screenshot Hasil Jawaban Soal 1 tampilan login



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



Gambar 25. Screenshot Hasil Jawban 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.

AppCompatDelegate: BarisAppCompatDelegate.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, productListType):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.

Konstruktor 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 AppCompat 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:listData 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 estetis, 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 estetis 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>