

LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE



Oleh:

Sheila Sabina NIM. 2310817220028

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
JUNI 2025**

LEMBAR PENGESAHAN

LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Sheila Sabina
NIM : 2310817220028

Menyetujui,
Asisten Praktikum
Mengetahui,
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026 NIP. 198810272019032013

DAFTAR ISI

LEMBAR PENGESAHAN.....	2
DAFTAR ISI	3
DAFTAR GAMBAR.....	5
DAFTAR TABEL	6
MODUL 1 : Android Basic with Kotlin.....	9
SOAL 1	9
A. Source Code	11
B. Output Program.....	16
C. Pembahasan.....	17
MODUL 2 : Android Layout.....	23
SOAL 1	23
A. Source Code	24
B. Output Program.....	32
C. Pembahasan.....	33
MODUL 3 : Build a Scrollable List	40
SOAL 1	40
A. Source Code	42
B. Output Program.....	65
C. Pembahasan.....	67
MODUL 4 : ViewModel and Debugging.....	86
SOAL 1	86
A. Source Code	86
B. Output Program.....	113
C. Pembahasan.....	116
SOAL 2.....	141
MODUL 5 : Connect to the Internet.....	143
SOAL 1	143
A. Source Code	143
B. Output Program.....	182

C. Pembahasan.....	185
TAUTAN GIT	215

DAFTAR GAMBAR

Gambar 1. Screenshot Hasil Jawaban Soal 1 Modul 1	16
Gambar 2. Screenshot Hasil Jawaban Soal 1 Modul 2	33
Gambar 3. Screenshot Hasil Jawaban Soal 1Modul 3	67
Gambar 4. Screenshot Item List Gunung Hasil Jawaban Soal 1 Modul 4	115
Gambar 5. Debugger.....	115
Gambar 6. Step Over (F8)	115
Gambar 7. Step Into (F7).....	116
Gambar 8. Step Out (Shift+F8)	116
Gambar 10. Screesnshot Output Program Soal 1	184

DAFTAR TABEL

Tabel 1. Source Code Soal 1 Modul 1.....	11
Tabel 2. Source Code Soal 1 Modul 1.....	13
Tabel 3. Source Code Soal 1 Modul 2.....	24
Tabel 4. Source Code Soal 1 Modul 2.....	26
Tabel 5. Source Code Soal 1 Modul 2.....	30
Tabel 6. Source Code Soal 1 Modul 3.....	42
Tabel 7. Source Code Gunung.kt.....	43
Tabel 8. Source Code GunungAdapter.kt.....	44
Tabel 9. Source Code ListFragment.kt.....	45
Tabel 10. Source Code MainActivity.kt.....	48
Tabel 11. Source Code activity_main.xml	49
Tabel 12. Source Code detail_fragment.xml	49
Tabel 13. Source Code item_gunung.xml	52
Tabel 14. Source Code list_fragment.xml	55
Tabel 15. Source Code colors.xml.....	56
Tabel 16. Source Code string.xml	56
Tabel 17. Source Code theme.xml.....	60
Tabel 18. Source Code AndroidManifest.xml.....	61
Tabel 19. Source Code BuildGradle.kts	62
Tabel 20. Source Code Soal 1 Modul 4.....	86
Tabel 21. Source Code Gunung.kt.....	88
Tabel 22. Source Code Gunung.kt Soal 1 Modul 4.....	88
Tabel 23. Source Code GunungViewModel.kt Soal 1 Modul 4.....	90
Tabel 24. Source Code GunungViewModelFactory.kt Soal 1 Modul 4	92
Tabel 25. Source Code ListFragment.kt Soal 1 Modul 4	92
Tabel 26. Source Code MainActivity.kt Soal 1 Modul 4	96
Tabel 27. Source Code activity_main.xml Soal 1 Modul 4.....	97
Tabel 28. Source Code detail_fragment.xml Soal 1 Modul 4	97
Tabel 29. Source Code item_gunung.xml Soal 1 Modul 4	100
Tabel 30. Source Code list_fragment.xml Soal 1 Modul 4	103

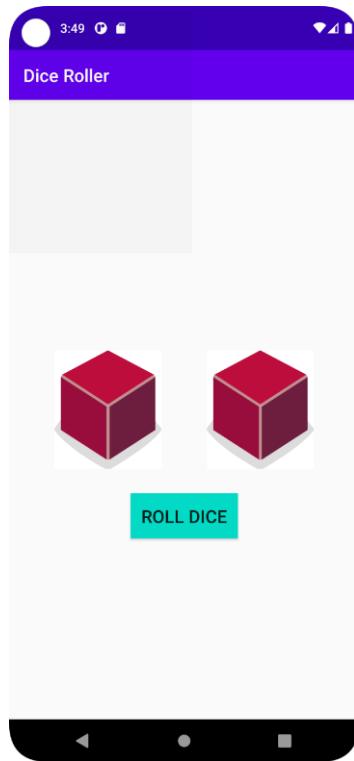
Tabel 31. Source Code colors.xml.....	104
Tabel 32. Source Code string.xml Soal 1 Modul 4	104
Tabel 33. Source Code theme.xml Soal 1 Modul 4.....	108
Tabel 34. Source Code AndroidManifest.xml Soal 1 Modul 4	109
Tabel 35. Source Code BuildGradle.kts Soal 1 Modul 4	110
Tabel 36. Source Code AndroidManifest.xml.....	143
Tabel 37. Source Code GunungDao	144
Tabel 38. Source Code AppDatabase	146
Tabel 39. Source Code GunungEntity.kt.....	147
Tabel 40. Source Code GunungMapper.kt	147
Tabel 41. Source Code Gunung.....	148
Tabel 42. Source Code ApiClient.....	149
Tabel 43. Source Code GunungApiService.....	150
Tabel 44. Source Code GunungRepository	151
Tabel 45. Source Code RepositoryInstance.....	153
Tabel 46. Source Code FavoriteGunungFragment.kt.....	153
Tabel 47. Source Code GunungAdapter.....	156
Tabel 48. Source Code GunungDetailFragment.....	158
Tabel 49. Source Code GunungListFragment	160
Tabel 50. Source Code ViewPagerAdapter.....	163
Tabel 51. Source Code GunungViewModel.....	164
Tabel 52. Source Code GunungViewModelFactory	167
Tabel 53. Source Code MainActivity	168
Tabel 54. Source Code activity_main.xml	168
Tabel 55. Source Code detail_fragment.xml	170
Tabel 56. Source Code fragment_favorit_gunung.xml	172
Tabel 57. Source Code item_gunung.xml	172
Tabel 58. Source Code list_fragment.xml	177
Tabel 59. Source Code colors.xml.....	177
Tabel 60. Source Code string.xml	178
Tabel 61. Source Code themes.xml	178

MODUL 1 : Android Basic with Kotlin

SOAL 1

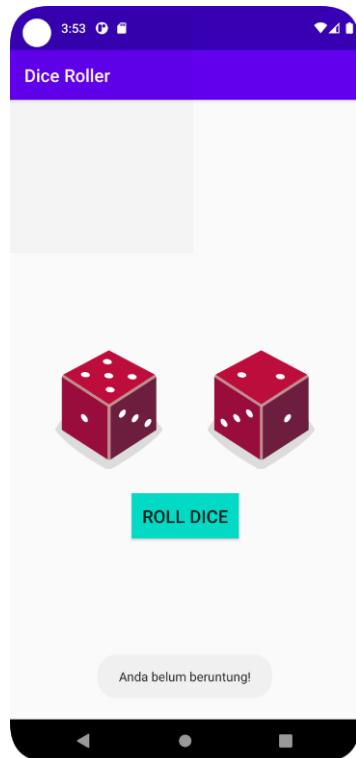
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



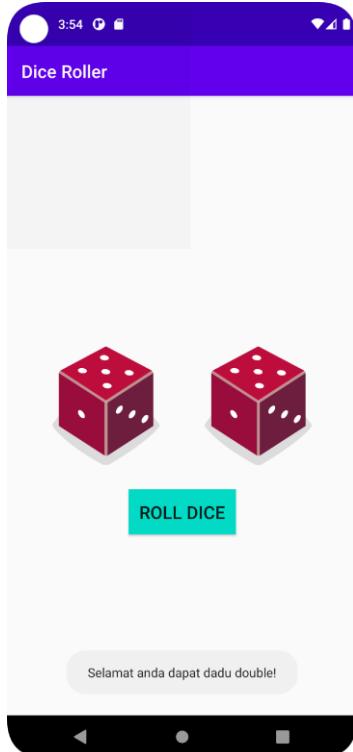
Gambar 1 Tampilan Awal Aplikasi

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



Gambar 2 Tampilan Dadu Setelah Di Roll

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan “Selamat anda dapat dadu double!” seperti dapat dilihat pada Gambar 3.
4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project**. Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.
5. Untuk gambar dadu dapat didownload pada link berikut:
https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export=download



Gambar 3 Tampilan Roll Dadu Double

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Soal 1 Modul 1

1	package	com.example.dice_roller
2		
3	import	android.os.Bundle
4	import	android.widget.Button
5	import	android.widget.ImageView
6	import	android.widget.Toast
7	import	androidx.activity.enableEdgeToEdge
8	import	androidx.appcompat.app.AppCompatActivity
9	import	androidx.core.view.ViewCompat
10	import	androidx.core.view.WindowInsetsCompat
11	com.example.dice_roller.databinding.ActivityMainBinding	

```
12
13 class MainActivity : AppCompatActivity() {
14
15     private lateinit var binding: ActivityMainBinding
16
17     override fun onCreate(savedInstanceState: Bundle?) {
18         super.onCreate(savedInstanceState)
19         enableEdgeToEdge()
20         binding = ActivityMainBinding.inflate(layoutInflater)
21         setContentView(binding.root)
22
23         // dice1 = findViewById(R.id.dice_image1)
24         // dice2 = findViewById(R.id.dice_image2)
25
26         // roll_button = findViewById(R.id.roll_button)
27
28         binding.rollButton.setOnClickListener {
29             rollDice()
30         }
31     }
32
33     private fun rollDice() {
34
35         val randomInt1 = (1..6).random()
36
37         val drawableResource1 = when (randomInt1) {
38             1 -> R.drawable.dice_1
39             2 -> R.drawable.dice_2
40             3 -> R.drawable.dice_3
41             4 -> R.drawable.dice_4
42             5 -> R.drawable.dice_5
43             6 -> R.drawable.dice_6
44             else -> R.drawable.dice_0
45         }
46
47         val randomInt2 = (1..6).random()
48
49         val drawableResource2 = when (randomInt2) {
```

```

43         1           ->          R.drawable.dice_1
44         2           ->          R.drawable.dice_2
45         3           ->          R.drawable.dice_3
46         4           ->          R.drawable.dice_4
47         5           ->          R.drawable.dice_5
48         6           ->          R.drawable.dice_6
49     else           ->          R.drawable.dice_0
50 }
51
52 binding.diceImage1.setImageResource(drawableResource1)
53
54
55     if      (randomInt1      ==      randomInt2)      {
56
57         Toast.makeText(this, "Selamat Kamu dapat
58 Double!", Toast.LENGTH_SHORT).show()
59     }           else           {
60
61         Toast.makeText(this, "Anda Kurang Beruntung",
62 Toast.LENGTH_SHORT).show()
63     }
64 }
65 }
```

2. activity_main.xml

Tabel 2. Source Code Soal 1 Modul 1

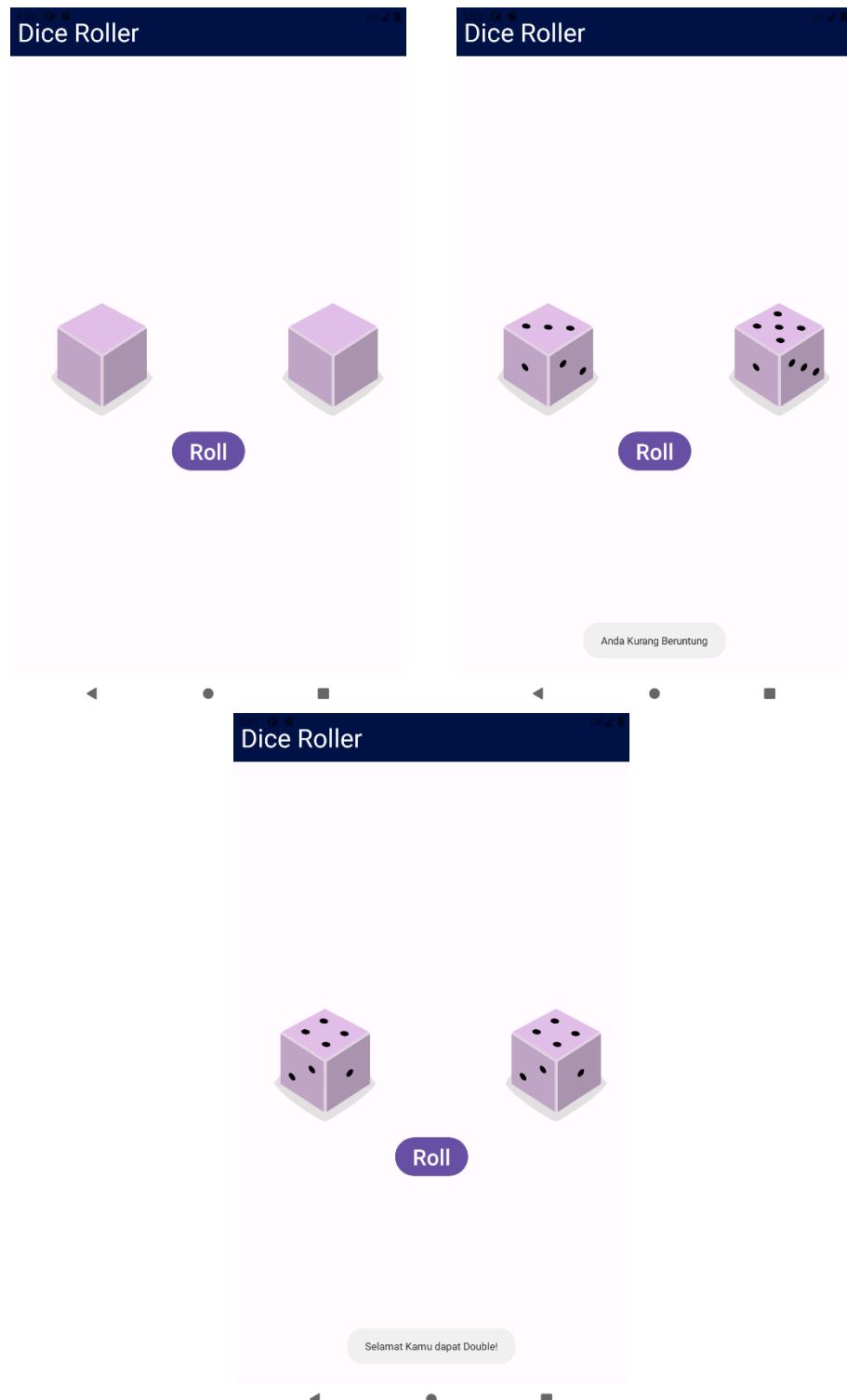
```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:id="@+id/main"
7     android:layout_width="match_parent"
8     android:layout_height="match_parent"
9     tools:context=".MainActivity">
```

```
9
10    <TextView
11        android:layout_width="match_parent"
12        android:layout_height="wrap_content"
13        android:text="Dice"                                     Roller"
14        android:textColor="@color/white"
15        android:textSize="35sp"
16        app:layout_constraintStart_toStartOf="parent"
17        app:layout_constraintTop_toTopOf="parent"
18        android:padding="10dp"
19        android:background="#001146"
20        />
21
22    <ImageView
23        android:id="@+id/dice_image1"
24        android:layout_width="200dp"
25        android:layout_height="200dp"
26        android:layout_marginStart="25dp"
27        android:src="@drawable/dice_0"
28        app:layout_constraintBottom_toBottomOf="parent"
29        app:layout_constraintStart_toStartOf="parent"
30        app:layout_constraintTop_toTopOf="parent"
31        />
32
33    <ImageView
34        android:id="@+id/dice_image2"
35        android:layout_width="200dp"
36        android:layout_height="200dp"
37        android:layout_marginStart="25dp"
38        android:src="@drawable/dice_0"
39        app:layout_constraintBottom_toBottomOf="parent"
40        app:layout_constraintEnd_toEndOf="parent"
41        app:layout_constraintTop_toTopOf="parent"
42        />
```

```
40
41     <Button
42         android:id="@+id/roll_button"
43         android:layout_width="wrap_content"
44         android:layout_height="wrap_content"
45         app:layout_constraintBottom_toBottomOf="parent"
46         app:layout_constraintEnd_toEndOf="parent"
47         app:layout_constraintStart_toStartOf="parent"
48         app:layout_constraintTop_toTopOf="parent"
49         android:layout_marginTop="250dp"
50         android:text="Roll"
51         android:textSize="30sp"
52
53     </androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

1. MainActivity.kt:

Pada baris [1] terdapat fungsi package com.example.dice_roller yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.dice_roller. Paket (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [2] hingga [11] terdapat fungsi import yang digunakan untuk mengimpor berbagai komponen yang dibutuhkan dalam pengembangan aplikasi Android. Baris import android.os.Bundle, android.widget.Button, android.widget.ImageView, dan android.widget.Toast digunakan untuk mengakses komponen dasar Android seperti tombol, gambar, dan pesan pop-up. Sementara itu, androidx.activity.enableEdgeToEdge, androidx.appcompat.app.AppCompatActivity, dan androidx.core.view digunakan untuk mendukung tampilan antarmuka yang lebih modern dan kompatibilitas dengan berbagai versi Android. Terakhir, com.example.dice_roller.databinding.ActivityMainBinding mengimpor class yang dihasilkan secara otomatis oleh View Binding untuk mengakses elemen-elemen UI di layout activity_main.xml dengan lebih efisien dan aman. Pada baris [13] hingga [28] terdapat fungsi bagian dari kelas MainActivity dalam aplikasi Android, yang berfungsi sebagai aktivitas utama saat aplikasi dijalankan. Kelas ini mewarisi AppCompatActivity, sehingga dapat menggunakan fitur-fitur kompatibilitas Android. Pada fungsi onCreate, yang dipanggil saat aktivitas pertama kali dibuat, terdapat beberapa hal penting seperti enableEdgeToEdge() yang berfungsi untuk mengatur agar tampilan aplikasi bisa menyesuaikan hingga ke tepi layar, memberikan tampilan yang lebih modern. binding = ActivityMainBinding.inflate(layoutInflater) yang digunakan untuk menginisialisasi objek *View Binding* untuk mengakses elemen-elemen UI tanpa perlu findViewById. setContentView(binding.root) yang digunakan untuk menentukan layout

yang akan ditampilkan di layar, yaitu root view dari file `activity_main.xml`.
`binding.rollButton.setOnClickListener { rollDice() }`
berfungsi untuk mengatur aksi saat tombol "roll" ditekan, yaitu memanggil fungsi `rollDice()` yang nantinya digunakan untuk melempar dadu.

Pada baris [31] hingga [52] terdapat fungsi `rollDice()` bertanggung jawab untuk mensimulasikan pelemparan dua buah dadu secara acak dan menentukan gambar dadu mana yang akan ditampilkan berdasarkan hasil tersebut. Di dalam fungsi ini, pertama-tama dibuat dua variabel `randomInt1` dan `randomInt2` yang masing-masing menghasilkan angka acak antara 1 hingga 6 menggunakan `(1..6).random()`. Angka ini merepresentasikan sisi dadu yang muncul. Setelah angka acak diperoleh, masing-masing angka tersebut dipetakan ke resource gambar (drawable) yang sesuai menggunakan struktur `when`. Misalnya, jika angka yang muncul adalah 3, maka gambar yang digunakan adalah `R.drawable.dice_3`. Jika secara tidak sengaja angka di luar 1–6 muncul (meskipun itu hampir tidak mungkin terjadi dengan range yang sudah ditentukan), maka `else` akan menampilkan gambar `default_dice_0`. Hasil dari pemetaan ini disimpan dalam `drawableResource1` dan `drawableResource2`, yang nantinya bisa digunakan untuk mengatur gambar pada dua `ImageView` dadu di tampilan aplikasi. Meskipun pada bagian kode ini belum ada perintah untuk menampilkan gambar ke tampilan (seperti `imageView.setImageResource(drawableResource1)`), namun logika inti untuk menentukan gambar yang sesuai berdasarkan angka acak sudah lengkap dan siap digunakan untuk animasi atau visualisasi dadu secara dinamis.

Baris [53] dan [54] terdapat fungsi yang digunakan untuk menyelesaikan proses simulasi lempar dadu dengan menampilkan gambar dadu di layar sesuai angka acak yang didapat, sehingga pengguna dapat melihat hasilnya secara visual. `binding.diceImage1` dan `binding.diceImage2` adalah referensi ke dua `ImageView` yang ada di layout XML, dan diakses menggunakan View Binding. `setImageResource(drawableResource1)` dan `setImageResource(drawableResource2)` digunakan untuk mengganti gambar di `ImageView` dengan gambar dadu yang sesuai berdasarkan angka acak

yang diperoleh sebelumnya. drawableResource1 dan drawableResource2 masing-masing berisi ID gambar (R.drawable.dice_1, dice_2, dst.) yang sesuai dengan hasil lemparan dadu. Pada baris [56] hingga [61] terdapat fungsi yang digunakan untuk memberikan umpan balik kepada pengguna berupa pesan pop up (Toast) berdasarkan hasil lemparan dua dadu. if (randomInt1 == randomInt2) berfungsi untuk mengecek apakah hasil dua dadu sama (misalnya keduanya 4 atau keduanya 6). Jika iya, maka kondisi dianggap "double". Toast.makeText(...).show() berguna untuk menampilkan pesan singkat di layar pengguna. Jika angkanya sama, maka muncul pesan: "Selamat Kamu dapat Double!" Jika berbeda, maka muncul pesan: "Anda Kurang Beruntung". this mengacu pada konteks MainActivity, yang dibutuhkan untuk menampilkan Toast. Toast.LENGTH_SHORT menentukan durasi tampilan Toast (sebentar saja).

2. activity_main.xml

Pada baris [1] hingga [9] terdapat fungsi yang merupakan kerangka dasar dari tampilan layar utama aplikasi android menggunakan ConstraintLayout, yang siap untuk diisi dengan elemen – elemen UI seperti tombol, gambar, teks, dan lainnya. <?xml version="1.0" encoding="utf-8"?> merupakan baris deklarasi XML, menunjukkan bahwa file ini ditulis dalam format XML versi 1.0 dengan encoding UTF-8. ConstraintLayout merupakan jenis layout dari Android Jetpack yang fleksibel untuk menyusun tampilan UI secara responsif dengan mengatur posisi tiap elemen berdasarkan constraint (batasan). xmlns:android, xmlns:app, dan xmlns:tools: adalah deklarasi namespace yang dibutuhkan agar atribut-atribut XML dikenali dan digunakan dengan benar oleh sistem Android dan tools Android Studio. android:id="@+id/main" yang akan memberikan ID pada layout ini, sehingga bisa diakses melalui kode Kotlin menggunakan View Binding atau findViewById. android:layout_width="match_parent" dan android:layout_height="match_parent" yang akan mengatur agar layout ini mengisi seluruh lebar dan tinggi layar.

tools:context=".MainActivity" yang akan memberitahu Android Studio bahwa layout ini digunakan dalam MainActivity, sehingga tampilan layout di editor dapat disesuaikan. Pada baris [11] hingga [21] terdapat fungsi yang berguna sebagai judul dari aplikasi dengan tampilan yang jelas dan mencolok di bagian atas layar, memberikan identitas visual bahwa aplikasi ini adalah "Dice Roller". android:layout_width="match_parent" digunakan untuk membuat lebar TextView mengikuti lebar parent (mengisi penuh dari kiri ke kanan). android:layout_height="wrap_content" berfungsi untuk membuat tinggi TextView menyesuaikan tinggi teks di dalamnya. android:text="Dice Roller" yang akan menentukan teks yang ditampilkan, dalam hal ini adalah "Dice Roller". android:textColor="@color/white" digunakan untuk mengatur warna teks menjadi putih, mengambil dari file colors.xml. android:textSize="35sp" yang berguna mengatur ukuran teks menjadi 35sp (scale-independent pixels), agar tetap terbaca dengan baik di berbagai ukuran layar. app:layout_constraintStart_toStartOf="parent" dan app:layout_constraintTop_toTopOf="parent" yang akan menempatkan TextView di bagian kiri atas layar, dengan constraint ke sisi kiri dan atas dari layout induk (ConstraintLayout). android:padding="10dp" yang akan memberikan ruang di dalam TextView sebesar 10dp dari semua sisi agar teks tidak terlalu menempel ke tepi. android:background="#001146" berguna untuk memberi latar belakang berwarna biru gelap (kode hex #001146).

Pada baris [23] hingga [31] terdapat fungsi yang digunakan untuk menampilkan gambar dadu di tengah layar. Gambar ini memiliki ukuran 200dp x 200dp, dengan jarak margin 25dp dari sisi kiri. Gambar dadu pertama (dice_image1) ditampilkan dengan gambar default dice_0 dan diposisikan di tengah layar dengan ConstraintLayout. android:id="@+id/dice_image1" berguna untuk memberikan ID unik untuk ImageView ini, yang dapat diakses di kode (misalnya melalui View Binding atau findViewById), untuk mengganti gambar atau mengatur propertiannya. android:layout_width="200dp" digunakan untuk menentukan lebar ImageView menjadi 200dp (density-independent

pixels). `android:layout_height="200dp"` yang berguna untuk menentukan tinggi ImageView menjadi `200dp`. `android:layout_marginStart="25dp"` untuk memberikan jarak margin sebesar `25dp` di sisi kiri dari ImageView, memberikan sedikit ruang antara gambar dan tepi layar. `android:src="@drawable/dice_0"` yang akan menetapkan gambar sumber yang ditampilkan di ImageView, dalam hal ini adalah gambar `dice_0`, yang kemungkinan besar merupakan gambar dadu kosong atau default. `app:layout_constraintBottom_toBottomOf="parent"` berguna untuk menempatkan bagian bawah ImageView pada bagian bawah dari parent layout (ConstraintLayout), memastikan gambar berada di bawah layar. `app:layout_constraintStart_toStartOf="parent"` yang akan menempatkan sisi kiri ImageView pada sisi kiri dari parent layout. `app:layout_constraintTop_toTopOf="parent"` yang akan menempatkan sisi atas ImageView pada sisi atas dari parent layout.

Pada baris [34] hingga [43] memiliki fungsi kode yang sama dengan blok kode yang sebelumnya namun dengan format gambar yang berbeda yakni gambar dadu pertama (`dice_image2`). Pada baris [45] hingga [56] terdapat fungsi button yang digunakan untuk memberikan interaksi pengguna, seperti tombol “roll” dadu. `android:id="@+id/roll_button"` berguna untuk memberikan ID unik untuk tombol ini, yang memungkinkan tombol tersebut diakses dalam kode melalui View Binding atau `findViewById`. `android:layout_width="wrap_content"` yang akan engatur lebar tombol agar hanya sebesar konten di dalamnya, yaitu teks “Roll”. `android:layout_height="wrap_content"` berfungsi untuk mengatur tinggi tombol agar hanya sebesar konten di dalamnya, sesuai dengan ukuran teks. `app:layout_constraintBottom_toBottomOf="parent"` berguna untuk menyusun tombol di bagian bawah layar, dengan mengikat bagian bawah tombol ke bagian bawah parent layout (ConstraintLayout). `app:layout_constraintEnd_toEndOf="parent"` yang akan menyusun tombol di sisi kanan layar, dengan mengikat sisi kanan tombol ke sisi kanan parent

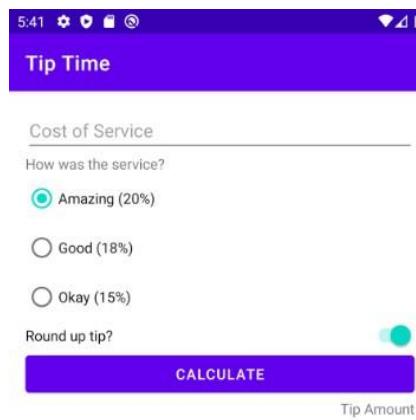
layout. `app:layout_constraintStart_toStartOf="parent"` yang berguna untuk menyusun tombol di sisi kiri layar, dengan mengikat sisi kiri tombol ke sisi kiri parent layout. `app:layout_constraintTop_toTopOf="parent"` yang akan menyusun tombol di bagian atas layar, dengan mengikat sisi atas tombol ke sisi atas parent layout. `android:layout_marginTop="250dp"` yang akan memberikan jarak margin sebesar 250dp di bagian atas tombol, memastikan tombol tidak terlalu dekat dengan bagian atas layar. `android:text="Roll"`: berguna untuk menentukan teks yang ditampilkan pada tombol, yaitu "Roll". `android:textSize="30sp"` berfungsi untuk menetapkan ukuran teks pada tombol menjadi 30sp (scale-independent pixels), memastikan teks tetap terbaca dengan baik di berbagai ukuran layar. Terakhir, pada baris [58] terdapat fungsi `</androidx.constraintlayout.widget.ConstraintLayout>` yang digunakan untuk sebagai tag penutup yang menandakan bahwa ConstraintLayout telah selesai dan semua elemen yang berada di dalamnya, seperti TextView, Button, dan ImageView, juga telah selesai didefinisikan. ConstraintLayout adalah jenis layout yang fleksibel dan efisien dalam menyusun tampilan UI di Android. Setiap elemen di dalam ConstraintLayout diatur dengan constraint (batasan) terhadap elemen lain atau terhadap parent layout.

MODUL 2 : Android Layout

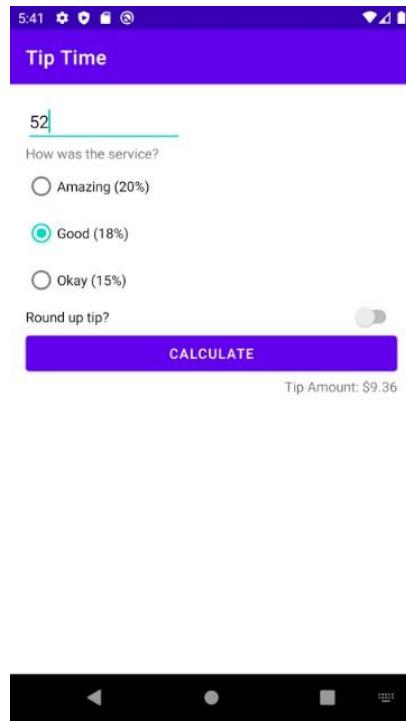
SOAL 1

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 Presentase Tip: Pengguna dapat memilih presentase 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 1 Tampilan Awal Aplikasi



Gambar 2 Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1. MainActivity.kt

Tabel 3. Source Code Soal 1 Modul 2

```
1 package com.example.androidlayout
2
3 import android.os.Bundle
4 import android.widget.*
5 import androidx.activity.viewModels
6 import androidx.appcompat.app.AppCompatActivity
7 import androidx.lifecycle.Observer
8
9 class MainActivity : AppCompatActivity() {
10
11     private val tipViewModel: TipViewModel by viewModels()
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
```

```
14     setContentView(R.layout.activity_main)
15
16     val etBiaya = findViewById<EditText>(R.id.etBiaya)
17
18     val rgTip = findViewById<RadioGroup>(R.id.rgTip)
19     val switchBulatkan = findViewById<Switch>(R.id.switchBulatkan)
20     val btnHitung = findViewById<Button>(R.id.btnHitung)
21
22     val tvHasil = findViewById<TextView>(R.id.tvHasil)
23
24
25     tipViewModel.biayaInput.observe(this) {
26         if (etBiaya.text.toString() != it)
27             etBiaya.setText(it)
28     }
29
30     tipViewModel.bulatkan.observe(this) {
31         switchBulatkan.isChecked = it
32     }
33
34     tipViewModel.tipResult.observe(this) {
35         tvHasil.text = it
36     }
37
38     btnHitung.setOnClickListener {
39         val biayaInput = etBiaya.text.toString()
40
41         if (biayaInput.isEmpty())
42             Toast.makeText(this, "Masukkan biaya layanan terlebih dahulu", Toast.LENGTH_SHORT).show()
43
44     }

```

```

45         val biaya = biayaInput.toDoubleOrNull()
46         if (biaya == null || biaya <= 0) {
47             Toast.makeText(this, "Biaya layanan harus
48 lebih dari 0", Toast.LENGTH_SHORT).show()
49             return@setOnClickListener
50         }
51
52         val persenTip = when
53             (rgTip.checkedRadioButtonId) {
54                 R.id.rb15 -> 0.15
55                 R.id.rb18 -> 0.18
56                 R.id.rb20 -> 0.20
57                 else -> {
58                     Toast.makeText(this,
59                         "Pilih
60 persentase tip", Toast.LENGTH_SHORT).show()
61                     return@setOnClickListener
62                 }
63             }
64
65         tipViewModel.setBiaya(biayaInput)
66         tipViewModel.setBulatkan(switchBulatkan.isChecked)
67         tipViewModel.setPersenTip(persenTip)
68         tipViewModel.hitungTip()
69     }

```

2. activity_main.xml

Tabel 4. Source Code Soal 1 Modul 2

1	<?xml version="1.0" encoding="utf-8"?>
2	<LinearLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	android:layout_width="match_parent"

```
5     android:layout_height="match_parent"
6     android:background="@android:color/white"
7     android:orientation="vertical">
8
9     <TextView
10        android:layout_width="match_parent"
11        android:layout_height="56dp"
12        android:background="#6200EE"
13        android:gravity="center_vertical"
14        android:paddingStart="16dp"
15        android:text="Tip Time"
16        android:textColor="@android:color/white"
17        android:textSize="20sp" />
18
19     <ScrollView
20        android:layout_width="match_parent"
21        android:layout_height="match_parent"
22        android:padding="16dp">
23
24         <LinearLayout
25            android:layout_width="match_parent"
26            android:layout_height="wrap_content"
27            android:orientation="vertical">
28
29             <EditText
30                android:id="@+id/etBiaya"
31                android:layout_width="match_parent"
32                android:layout_height="wrap_content"
33                android:backgroundTint="#6200EE"
34                android:hint="Cost of Service"
35                android:inputType="numberDecimal" />
36
37             <TextView
38                android:layout_width="wrap_content"
```

```
36         android:layout_height="wrap_content"
37         android:layout_marginTop="16dp"
38         android:text="How      was      the      service?"
39         android:textColor="#888888"                  />
40
41     <RadioGroup
42         android:id="@+id/rgTip"
43         android:layout_width="match_parent"
44         android:layout_height="wrap_content">
45
46         <RadioButton
47             android:id="@+id/rb20"
48             android:layout_width="wrap_content"
49             android:layout_height="wrap_content"
50             android:text="Amazing      (20%)"      />
51
52         <RadioButton
53             android:id="@+id/rb18"
54             android:layout_width="wrap_content"
55             android:layout_height="wrap_content"
56             android:text="Good      (18%)"      />
57
58         <RadioButton
59             android:id="@+id/rb15"
60             android:layout_width="wrap_content"
61             android:layout_height="wrap_content"
62             android:text="Okay      (15%)"      />
63
64     </RadioGroup>
65
66     <LinearLayout
67         android:layout_width="match_parent"
68         android:layout_height="wrap_content"
69         android:layout_marginTop="16dp"
70         android:orientation="horizontal">
```

```
67
68        <TextView
69            android:layout_width="0dp"
70            android:layout_height="wrap_content"
71            android:layout_weight="1"
72            android:text="Round      up      tip?"      />
73
74        <Switch
75            android:id="@+id/switchBulatkan"
76            android:layout_width="wrap_content"
77            android:layout_height="wrap_content"
78        />
79
80    </LinearLayout>
81
82
83
84
85
86
87
88 />
89
90
91        <Button
92            android:id="@+id/btnHitung"
93            android:layout_width="match_parent"
94            android:layout_height="wrap_content"
95            android:layout_marginTop="24dp"
96            android:backgroundTint="#6200EE"
97            android:text="CALCULATE"
98            android:textColor="@android:color/white"
99        />
100
101
102        <TextView
103            android:id="@+id/tvHasil"
104            android:layout_width="match_parent"
105            android:layout_height="wrap_content"
106            android:layout_marginTop="24dp"
107            android:gravity="end"
108            android:text="Tip           Amount"
109            android:textColor="#888888"
110            android:textSize="16sp"                  />
```

```
98     </LinearLayout>
99   </ScrollView>
100 </LinearLayout>
```

3. TipViewModel.kt

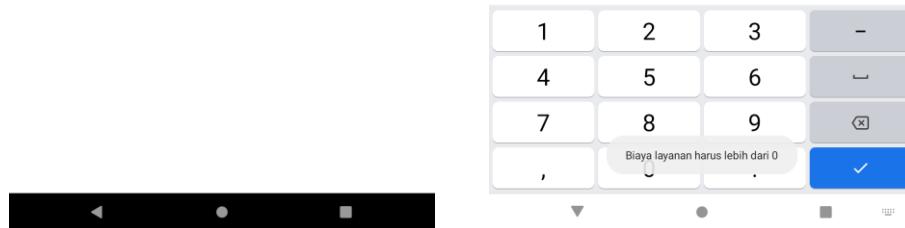
Tabel 5. Source Code Soal 1 Modul 2

```
1 package com.example.androidlayout
2
3 import androidx.lifecycle.LiveData
4 import androidx.lifecycle.MutableLiveData
5 import androidx.lifecycle.ViewModel
6 import kotlin.math.ceil
7
8 class TipViewModel : ViewModel() {
9
10    private val _biayaInput = MutableLiveData<String>()
11    val biayaInput: LiveData<String> = _biayaInput
12
13    private val _tipResult = MutableLiveData<String>()
14    val tipResult: LiveData<String> = _tipResult
15
16    private val _bulatkan = MutableLiveData<Boolean>()
17    val bulatkan: LiveData<Boolean> = _bulatkan
18
19    private var persenTip: Double = 0.0
20
21    fun setBiaya(biaya: String) {
22        _biayaInput.value = biaya
23    }
24
25    fun setBulatkan(value: Boolean) {
26        _bulatkan.value = value
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
```

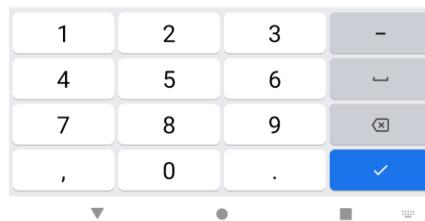
```
27     fun setPersenTip(value: Double) {
28         persenTip = value
29     }
30
31     fun hitungTip() {
32         val biaya = _biayaInput.value?.toDoubleOrNull()
33         if (biaya == null || biaya <= 0 || persenTip ==
34             0.0) {
35             _tipResult.value = ""
36             return
37         }
38         var tip = biaya * persenTip
39         if (_bulatkan.value == true) {
40             tip = ceil(tip)
41         }
42
43         _tipResult.value = "Jumlah tip: Rp
44             ${"% .2f".format(tip)}"
45     }
46 }
```

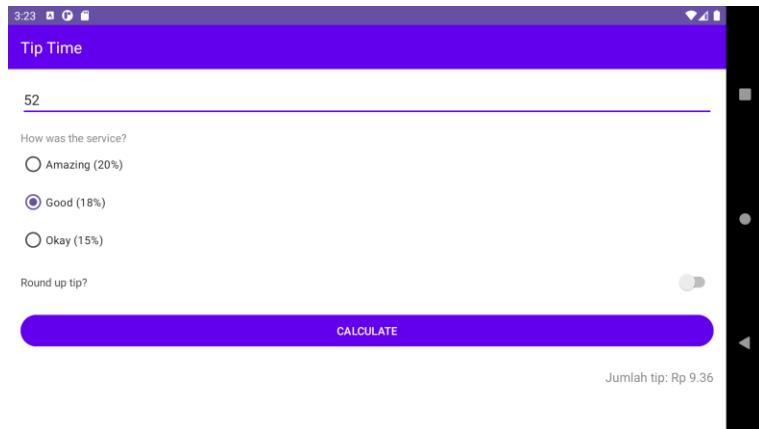
B. Output Program

The screenshots show the 'Tip Time' application interface. Both screens have a purple header bar with the title 'Tip Time'. The first screen has a text input field labeled 'Cost of Service' containing '100'. Below it is a radio button group for service quality: 'Amazing (20%)', 'Good (18%)', and 'Okay (15%)', with 'Good (18%)' selected. A switch labeled 'Round up tip?' is turned off. A blue 'CALCULATE' button is at the bottom. The second screen shows the same layout but with the text input field empty ('0').



The screenshot shows the 'Tip Time' application. The text input field now contains '100'. The radio button group for service quality shows 'Good (18%)' selected. The switch for rounding up the tip is turned off. The blue 'CALCULATE' button is at the bottom. The message 'Biaya layanan harus lebih dari 0' is no longer visible.





Gambar 2. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

1. MainActivity.kt:

Pada baris [1] terdapat fungsi package com.example.androidlayout yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.androidlayout. Paket (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [7] terdapat fungsi import yang digunakan untuk mengimpor berbagai komponen yang dibutuhkan dalam pengembangan aplikasi Android. Baris import android.os.Bundle, yang digunakan untuk mengakses kelas bundle yang biasanya dipakai dalam onCreate untuk menyimpan atau mengakses data yang dikirim saat Activity dibuat. android.widget.*., digunakan untuk mengimpor semua kelas yang ada di dalam sehingga bisa langsung menggunakan kelas itu tanpa harus ditulis lengkap. android.activiy.viewModel, digunakan untuk mengimpor fungsi ekstensi Kotlin ViewModels() dari jetpack lifecycle library yang berfungsi untuk mengambil instance ViewModel dengan cara yang aman terhadap lifecycle Activity atau Fragment. import androidx.appcompat.app.AppCompatActivity, digunakan untuk mengimpor kelas AppCompatActivity dari androidx, basis kelas untuk Activity yang mendukung fitur – fitur modern dan kompatibel dengan Android versi

lama. dan import androidx.lifecycle.Observer digunakan untuk mengamati perubahan LiveData. Digunakan saat ingin update UI secara otomatis saat data di ViewModel berubah. Pada baris [9] terdapat fungsi kelas MainActivity yang mewarisi AppCompatActivity yang merupakan class dasar untuk Activity dengan dukungan kompatibilitas ke Android versi lama.

Pada baris [11] terdapat fungsi private val tipViewModel: TipViewModel by viewModels() yang akan digunakan untuk menyimpan dan mengelola data aplikasi, serta delegasi dari Jetpack Lifecycle secara otomatis untuk menjaga data tetap bertahan saat rotasi layar dan sifat nya private, artinya hanya variabel hanya bisa diakses di dalam MainActivity. Pada baris [13] hingga [15] terdapat fungsi override fun onCreate(savedInstanceState: Bundle?) digunakan sebagai tempat mulai hidupnya Activity dan semua setup awal dilakukan dan untuk menyimpan dan memulihkan data saat terjadi perubahan konfigurasi (misalnya rotasi layar), sehingga data sebelumnya tidak hilang. Fungsi super.onCreate(savedInstanceState) digunakan untuk memastikan lifecycle bawaan Activity berjalan dengan benar. Fungsi setContentView(R.layout.activity_main) digunakan untuk menampilkan layout tampilan (UI) dari file XML ke dalam Activity. Pada baris [17] hingga [22] terdapat fungsi yang digunakan untuk menghubungkan komponen UI di layout XML ke variabel di dalam kode Kotlin. Setiap findViewById mencari elemen berdasarkan ID-nya, sehingga bisa digunakan untuk membaca input, menampilkan hasil, atau menangani interaksi. Fungsi ini menginisialisasi semua view (EditText, RadioGroup, Switch, Button, dan TextView) agar bisa digunakan dalam logika aplikasi. Pada baris [24] hingga [35] terdapat fungsi kode yang menggunakan LiveData Observer untuk memantau perubahan data di ViewModel. Saat data berubah, UI akan otomatis diperbarui. biayaInput berguna untuk mengatur nilai EditText agar sesuai dengan data terbaru. Bulatkan digunakan untuk menyesuaikan status Switch (on/off). tipResult digunakan untuk menampilkan hasil tip ke TextView. Sehingga fungsi kode ini membuat UI selalu sinkron dengan data di ViewModel secara otomatis. Pada baris [37] hingga [60] terdapat fungsi kode logika, dimana tombol "Hitung" ditekan, akan berfungsi untuk mengambil input biaya dari

pengguna, validasi input (tidak kosong, angka, dan lebih dari 0) serta menentukan persentase tip yang dipilih dari RadioButton. Jika ada yang tidak sesuai (input kosong, salah, atau belum pilih tip), akan muncul pesan kesalahan (Toast). Jadi, kode ini memproses input dan validasi sebelum menghitung tip. Pada baris [64] hingga [67] terdapat fungsi kode yang akan mengatur nilai-nilai yang diperlukan untuk menghitung tip, seperti biaya total, opsi pembulatan, dan persentase tip, kemudian memanggil fungsi untuk menghitung tip berdasarkan parameter tersebut. Semua pengaturan dilakukan dalam objek tipViewModel.

1. activity_main.xml

Pada baris [1] terdapat fungsi <?xml version="1.0" encoding="utf-8"?> yang digunakan untuk deklarasi XML yang memberi tahu bahwa file ini adalah file XML dan menggunakan versi XML 1.0 serta encoding karakter UTF-8. Ini merupakan bagian penting dari setiap file XML untuk memastikan interpretasi yang benar terhadap isi file, terutama dalam hal karakter yang digunakan. Pada baris [2] hingga [7] terdapat fungsi yang mendefinisikan sebuah LinearLayout di XML untuk tampilan Android. xmlns:android="http://schemas.android.com/apk/res/android" digunakan untuk mendeklarasikan namespace Android yang diperlukan untuk atribut XML Android. android:layout_width="match_parent" digunakan untuk menetapkan lebar LinearLayout agar sesuai dengan lebar layar induk (parent). android:layout_height="match_parent" untuk menetapkan tinggi LinearLayout agar sesuai dengan tinggi layar induk (parent). android:background="@android:color/white" untuk memberikan latar belakang putih pada LinearLayout. android:orientation="vertical" untuk mengatur orientasi LinearLayout secara vertikal, yang berarti elemen-elemen di dalamnya akan ditata secara vertikal (atas ke bawah). Sehingga, kode ini membuat sebuah LinearLayout dengan latar belakang putih dan elemen-elemen di dalamnya disusun secara vertikal. Pada baris [9] hingga [17] terdapat fungsi kode TextView, dengan lebar yang menyesuaikan induknya dan tinggi 56dp. Latar belakang <TextView berwarna ungu (#6200EE),

dengan teks "Tip Time" yang berwarna putih dan ukuran 20sp. Teks ditempatkan secara vertikal di tengah, dengan padding 16dp di sisi kiri (start), memberikan tampilan yang rapi dan teratur. Pada baris [19] hingga [22] terdapat fungsi <ScrollView yang mendefinisikan dengan lebar dan tinggi sesuai dengan ukuran induknya (match_parent). ScrollView ini juga diberi padding sebesar 16dp di semua sisi, sehingga konten di dalamnya tidak menempel langsung ke tepi layar. Dengan penggunaan ScrollView, elemen-elemen di dalamnya dapat digulir secara vertikal jika melebihi ukuran tampilan layar. Pada baris [24] hingga [27] terdapat fungsi <LinearLayout yang mendefinisikan dengan lebar yang sesuai dengan lebar induknya (match_parent) dan tinggi yang disesuaikan dengan kontennya (wrap_content). Orientasi LinearLayout ini diatur secara vertikal, yang berarti elemen-elemen di dalamnya akan disusun secara berurutan dari atas ke bawah.

Pada baris [29] hingga [35] terdapat fungsi >EditText yang digunakan untuk memasukkan teks, dengan beberapa pengaturan sebagai berikut: lebar EditText disesuaikan dengan lebar induknya (match_parent) dan tinggi disesuaikan dengan kontennya (wrap_content). EditText ini memiliki tint latar belakang berwarna ungu (#6200EE), menampilkan hint "Cost of Service" saat kosong, dan hanya memungkinkan input berupa angka desimal (inputType="numberDecimal"). ID yang diberikan adalah @+id/etBiaya, yang memungkinkan referensi elemen ini di kode Java atau Kotlin. Pada baris [37] hingga [42] terdapat kode fungsi <TextView yang mendefinisikan dengan lebar dan tinggi yang disesuaikan dengan kontennya (wrap_content). TextView ini memiliki margin atas sebesar 16dp, memberikan jarak antara elemen di atasnya. Teks yang ditampilkan adalah "How was the service?", dengan warna teks abu-abu (#888888). TextView ini biasanya digunakan untuk memberikan pertanyaan atau informasi kepada pengguna. Pada baris [44] hingga [47] terdapat fungsi kode <RadioGroup yang mendefinisikan dengan ID @+id/rgTip yang memiliki lebar sesuai dengan lebar induknya (match_parent) dan tinggi yang disesuaikan dengan kontennya (wrap_content). RadioGroup digunakan untuk mengelompokkan beberapa elemen RadioButton sehingga hanya satu pilihan yang dapat dipilih pada suatu waktu. Kode ini mengatur kontainer untuk elemen-elemen pilihan tip yang akan disediakan dalam aplikasi. Pada baris [49] hingga [65] terdapat

fungsi kode <RadioButton yang digunakan untuk mendefinisikan tiga RadioButton dalam sebuah RadioGroup, yang memungkinkan pengguna untuk memilih satu opsi dari beberapa pilihan yang ada. Setiap RadioButton memiliki ID unik (seperti @+id/rb20, @+id/rb18, dan @+id/rb15) serta lebar dan tinggi yang disesuaikan dengan kontennya (wrap_content). Teks yang ditampilkan pada masing-masing RadioButton adalah "Amazing (20%)", "Good (18%)", dan "Okay (15%)", yang menunjukkan persentase tip yang dapat dipilih pengguna. Ketika salah satu dipilih, pilihan lainnya secara otomatis akan terhapus. Pada baris [68] hingga [84] terdapat fungsi kode <LinearLayout yang didalamnya berisi <TextView dan <Switch, dengan tujuan memberikan opsi kepada pengguna untuk membulatkan tip. LinearLayout ini memiliki lebar penuh (match_parent), tinggi menyesuaikan kontennya (wrap_content), margin atas 16dp, dan orientasi horizontal agar kedua elemen ditampilkan sejajar secara mendatar. TextView menampilkan teks "Round up tip?" dan menggunakan layout_weight="1" serta layout_width="0dp" agar mengisi ruang kosong yang tersedia. Switch dengan ID @+id/switchBulatkan berada di sebelah kanan, memungkinkan pengguna untuk mengaktifkan atau menonaktifkan opsi pembulatan tip. Pada baris [87] hingga [94] terdapat fungsi <Button yang digunakan untuk sebuah LinearLayout horizontal yang berisi TextView dan Switch, dengan tujuan memberikan opsi kepada pengguna untuk membulatkan tip. LinearLayout ini memiliki lebar penuh (match_parent), tinggi menyesuaikan kontennya (wrap_content), margin atas 16dp, dan orientasi horizontal agar kedua elemen ditampilkan sejajar secara mendatar. TextView menampilkan teks "Round up tip?" dan menggunakan layout_weight="1" serta layout_width="0dp" agar mengisi ruang kosong yang tersedia. Switch dengan ID @+id/switchBulatkan berada di sebelah kanan, memungkinkan pengguna untuk mengaktifkan atau menonaktifkan opsi pembulatan tip. Pada baris [96] hingga [104] terdapat fungsi <TextView yang digunakan untuk mendefinisikan dengan ID @+id/tvHasil yang memiliki lebar penuh (match_parent) dan tinggi menyesuaikan kontennya (wrap_content). TextView ini diberi margin atas sebesar 24dp, teksnya "Tip Amount", warna teks abu-abu (#888888), dan ukuran teks 16sp. Properti android:gravity="end" membuat teks diratakan ke kanan. TextView ini berfungsi untuk menampilkan hasil perhitungan tip

kepada pengguna. Sehingga keseluruhan dari kode ini berguna untuk mendesain antarmuka aplikasi kalkulator tip di Android, yang terdiri dari input biaya layanan, pilihan persentase tip (melalui RadioButton), opsi pembulatan tip (dengan Switch), tombol untuk menghitung tip, dan TextView untuk menampilkan hasilnya. Seluruh elemen disusun secara vertikal dalam LinearLayout, dengan ScrollView agar tampilan tetap dapat digulir jika kontennya melebihi layar.

2. TipViewModel

Pada baris [1] terdapat fungsi package com.example.androidlayout yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.androidlayout. Paket (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [6] terdapat fungsi import yang digunakan untuk mengimpor berbagai komponen yang dibutuhkan dalam pengembangan aplikasi Android. Baris import androidx.lifecycle.LiveData digunakan untuk menyimpan data yang bisa diamati (observable) dan akan otomatis diperbarui saat nilainya berubah. import androidx.lifecycle.MutableLiveData digunakan untuk mengatur atau memperbarui data agar keamanannya terjaga. import androidx.lifecycle.ViewModel digunakan untuk memisahkan logic dan tampilan, dan menjaga data tetap hidup saat rotasi layar. import kotlin.math.ceil digunakan untuk membulatkan angka ke atas ke bilangan bulat terdekat, digunakan saat ingin pembulatan hasil perhitungan tip ke atas. Pada baris [8] terdapat fungsi class TipViewModel: ViewModel() yang digunakan mendefinisikan kelas untuk menyimpan dan mengelola data untuk UI, secara terpisah dari lifecycle UI. Pada baris [10] hingga [19] terdapat beberapa fungsi private yang berfungsi untuk menyimpan dan mengelola data internal yang dibutuhkan dalam progress perhitungan tip, yaitu _biayaInput digunakan untuk menyimpan input biaya dari pengguna. _tipResult digunakan untuk menyimpan hasil perhitungan tip yang akan ditampilkan ke UI. _bulatkan digunakan untuk menyimpan status apakah hasil tip perlu dibulatkan ke atas atau tidak. persentTip digunakan untuk menyimpan nilai persentase tip yang akan digunakan dalam

perhitungan. Semua data ini bersifat privat agar tidak bisa diubah langsung dari luar ViewModel, dan hanya bisa diakses secara aman melalui versi LiveData yang publik. Pada baris [21] hingga [46] terdapat fungsi setter yang digunakan untuk mengatur nilai input dari pengguna ke dalam variabel privat MutableLiveData atau variabel biasa. `fun setBiaya(biaya: String)` digunakan untuk mengatur nilai biaya yang dimasukkan pengguna. `fun setBulatkan(value: Boolean)` digunakan untuk mengatur apakah hasil tip perlu dibulatkan ke atas. `fun setPersentTip(value: Double)` digunakan untuk mengatur nilai persentase tip yang digunakan saat menghitung. fungsi `itungTip()`, digunakan untuk menghitung jumlah tip berdasarkan input pengguna. Dengan langkah langkah mengambil nilai biayaInput dan mengubahnya ke Double, mengecek apakah input valid (tidak kosong, tidak nol, dan persen tip tidak nol), menghitung `tip = biaya * persentTip`, jika opsi bulatkan aktif, hasil tip dibulatkan ke atas (`ceil`) serta menyimpan hasil ke `_tipResult` dalam format "Jumlah tip: Rp xx.xx"

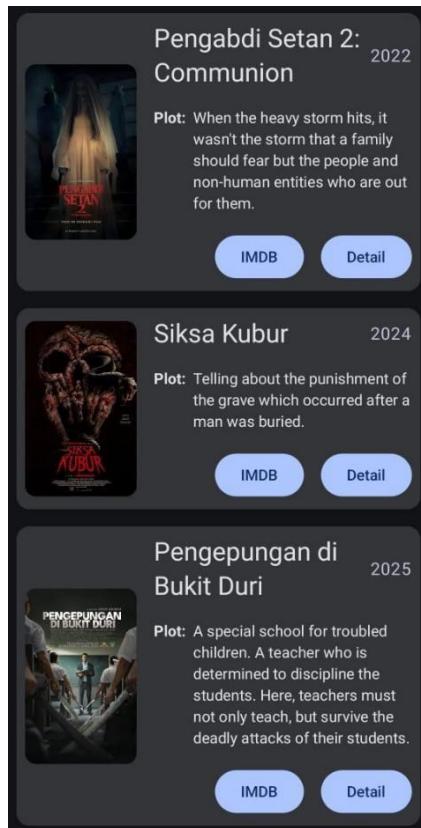
MODUL 3 : Build a Scrollable List

SOAL 1

Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
 2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
 3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4.
- Terdapat 2 button dalam list, dengan fungsi berikut:
- a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
 - b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
 6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
 7. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
 8. Aplikasi berbasis XML harus menggunakan ViewBinding

UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 1. Contoh UI List

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:



Gambar 2. Contoh UI Detail

A. Source Code

app\src\main\java\com\example\modul3

1. DetailFragment.kt

Tabel 6. Source Code Soal 1 Modul 3

```
1 package com.example.modul3
2
3 import android.os.Bundle
4 import androidx.fragment.app.Fragment
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.view.ViewGroup
8 import com.example.modul3.databinding.DetailFragmentBinding
9
10 class DetailFragment : Fragment() {
11
12     private var _binding: DetailFragmentBinding? = null
13     private val binding get() = _binding!!
14
15     override fun onCreateView(
16         inflater: LayoutInflater, container: ViewGroup?,
17         savedInstanceState: Bundle?
18     ): View {
19         _binding =
20             DetailFragmentBinding.inflate(inflater, container, false)
21
22         val image = arguments?.getInt("EXTRA_PHOTO")
23         val name = arguments?.getString("EXTRA_NAME")
24         val lokasi = arguments?.getString("EXTRA_LOKASI")
25         val deskripsi =
26             arguments?.getString("EXTRA_DESKRIPSI")
27 }
```

```

28     binding.tvName.text = name
29     binding.tvLokasi.text = lokasi
30     binding.tvDeskripsi.text = deskripsi
31     image?.let {
32         binding.imgPoster.setImageResource(it)
33     }
34     return binding.root
35 }
36
37 override fun onDestroyView() {
38     super.onDestroyView()
39     _binding = null
40 }
41 }
```

2. Gunung.kt

Tabel 7. Source Code Gunung.kt

```

1 package com.example.modul3
2
3 import android.os.Parcelable
4 import kotlinx.parcelize.Parcelize
5
6 @Parcelize
7 data class Gunung(
8     val image: Int,
9     val name: String,
10    val lokasi: String,
11    val deskripsi: String,
12    val link: String
13
14
15 ): Parcelable
```

3. GunungAdapter.kt

Tabel 8. Source Code GunungAdapter.kt

```
1 package com.example.modul3
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.modul3.databinding.ItemGunungBinding
7
8 class GunungAdapter(
9     private val listGunung: ArrayList<Gunung>,
10    private val onLinkClick: (String) -> Unit,
11    private val onDetailClick: (Int, String, String, String) ->
12 Unit
13 ) : RecyclerView.Adapter<GunungAdapter.ListViewHolder>() {
14
15     class ListViewHolder(private val binding:
16 ItemGunungBinding) : RecyclerView.ViewHolder(binding.root) {
17         fun bind(gunung: Gunung, onLinkClick: (String) -> Unit,
18 onDetailClick: (Int, String, String, String) -> Unit) {
19             // Bind data to views using the binding object
20             binding.tvGunungName.text = gunung.name
21             binding.tvGunungLokasi.text = gunung.lokasi
22             binding.tvGunungDeskripsi.text = gunung.deskripsi
23             binding.imgGunung.setImageResource(gunung.image)
24
25             binding.btnLink.setOnClickListener {
26                 onLinkClick(gunung.link) }
27
28             binding.btnDetail.setOnClickListener {
29                 onDetailClick(gunung.image, gunung.name,
30 gunung.lokasi, gunung.deskripsi)
31         }
```

```

32         }
33     }
34
35     override fun onCreateViewHolder(parent: ViewGroup,
36 viewType: Int): ListViewHolder {
37         val binding =
38             ItemGunungBinding.inflate(LayoutInflater.from(parent.context),
39             parent, false)
40         return ListViewHolder(binding)
41     }
42
43     override fun getItemCount(): Int = listGunung.size
44
45     override fun onBindViewHolder(holder: ListViewHolder,
46 position: Int) {
47         val gunung = listGunung[position]
48         holder.bind(gunung, onLinkClick, onDetailClick)
49     }
50 }
```

4. ListFragment.kt

Tabel 9. Source Code ListFragment.kt

```

1 package com.example.modul3
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.os.Bundle
6 import androidx.fragment.app.Fragment
7 import android.view.LayoutInflater
8 import android.view.View
9 import android.view.ViewGroup
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import com.example.modul3.databinding.ListFragmentBinding
```

```
12
13 class ListFragment : Fragment() {
14
15     private var _binding: ListFragmentBinding? = null
16     private val binding get() = _binding!!
17
18     private lateinit var gunungAdapter: GunungAdapter
19     private val list = ArrayList<Gunung>()
20
21     override fun onCreateView(
22         inflater: LayoutInflater, container: ViewGroup?,
23         savedInstanceState: Bundle?
24     ): View {
25         _binding = ListFragmentBinding.inflate(inflater,
26         container, false)
27
28         list.clear()
29         list.addAll(getListGunung())
30         setupRecyclerView()
31
32         return binding.root
33     }
34
35     private fun setupRecyclerView() {
36         gunungAdapter = GunungAdapter(
37             list,
38             onLinkClick = { link ->
39                 val intent = Intent(Intent.ACTION_VIEW,
40                 Uri.parse(link))
41                 startActivity(intent)
42             },
43             onDetailClick = { image, name, lokasi, deskripsi ->
44                 val detailFragment = DetailFragment().apply {
45                     arguments = Bundle().apply {
```

```
46                     putInt("EXTRA_PHOTO", image)
47                     putString("EXTRA_NAME", name)
48                     putString("EXTRA_LOKASI", lokasi)
49                     putString("EXTRA_DESKRIPSI", deskripsi)
50                 }
51             }
52             parentFragmentManager.beginTransaction()
53             .replace(R.id.frame_container,
54 detailFragment)
55             .addToBackStack(null)
56             .commit()
57         }
58     )
59
60     binding.rvGunung.apply {
61         layoutManager = LinearLayoutManager(context)
62         adapter = gunungAdapter
63         setHasFixedSize(true)
64     }
65 }
66
67     private fun getListGunung(): ArrayList<Gunung> {
68         val dataImage =
69 resources.obtainTypedArray(R.array.gunung_image)
70         val dataName =
71 resources.getStringArray(R.array.gunung_name)
72         val dataLokasi =
73 resources.getStringArray(R.array.gunung_lokasi)
74         val dataDesc =
75 resources.getStringArray(R.array.gunung_deskripsi)
76         val dataLink =
77 resources.getStringArray(R.array.gunung_link)
78
79 }
```

```

80         val listGunung = ArrayList<Gunung>()
81         for (i in dataName.indices) {
82             val gunung = Gunung(dataImage.getResourceId(i, -
83 1), dataName[i], dataLokasi[i], dataDesc[i], dataLink[i])
84             listGunung.add(gunung)
85         }
86         dataImage.recycle()
87         return listGunung
88     }
89
90     override fun onDestroyView() {
91         super.onDestroyView()
92         _binding = null
93     }
94 }
95
96

```

5. MainActivity.kt

Tabel 10. Source Code MainActivity.kt

```

1 package com.example.modul3
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5
6
7 class MainActivity : AppCompatActivity() {
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11
12        val fragmentManager = supportFragmentManager
13        val listFragment = ListFragment()

```

```

14     val fragment =
15     fragmentManager.findFragmentByTag(ListFragment::class.java.simpleName)
16
17     if (fragment !is ListFragment) {
18         fragmentManager
19             .beginTransaction()
20             .add(R.id.frame_container, listFragment,
21 ListFragment::class.java.simpleName)
22             .commit()
23     }
24 }
25 }
```

app\src\main\res\layout

6. activity_main.xml

Table 11. Source Code activity_main.xml

```

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

7. detail_fragment.xml

Table 12. Source Code detail_fragment.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
```

```
4      xmlns:tools="http://schemas.android.com/tools"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="#B6D6F1"
9      android:padding="16dp"
10     tools:context=".DetailFragment">
11
12     <androidx.constraintlayout.widget.ConstraintLayout
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content">
15
16         <ImageView
17             android:id="@+id/imgPoster"
18             android:layout_width="0dp"
19             android:layout_height="300dp"
20             android:contentDescription="Foto Gunung"
21             app:layout_constraintTop_toTopOf="parent"
22             app:layout_constraintStart_toStartOf="parent"
23             app:layout_constraintEnd_toEndOf="parent" />
24
25         <TextView
26             android:id="@+id/tvName"
27             android:layout_width="0dp"
28             android:layout_height="wrap_content"
29             android:gravity="center"
30             android:textSize="35sp"
31             android:textStyle="bold"
32             android:textColor="@android:color/black"
33             tools:text="Gunung Semeru"
34             android:layout_marginTop="12dp"
35             app:layout_constraintTop_toBottomOf="@+id/imgPoster"
36             app:layout_constraintStart_toStartOf="parent"
37             app:layout_constraintEnd_toEndOf="parent" />
```

```
38
39     <TextView
40         android:id="@+id/tvLokasi"
41         android:layout_width="0dp"
42         android:layout_height="wrap_content"
43         android:gravity="center"
44         android:textSize="22sp"
45         tools:text="Lokasi: Jawa Timur"
46         android:layout_marginTop="8dp"
47         app:layout_constraintTop_toBottomOf="@+id/tvName"
48         app:layout_constraintStart_toStartOf="parent"
49         app:layout_constraintEnd_toEndOf="parent" />
50
51     <TextView
52         android:id="@+id/tvDeskripsi"
53         android:layout_width="0dp"
54         android:layout_height="wrap_content"
55         android:textSize="18sp"
56         android:justificationMode="inter_word"
57         tools:text="Gunung tertinggi di Pulau Jawa yang
58 terkenal dengan jalur pendakian yang menantang dan keindahan
59 pemandangan alam."
60         android:layout_marginTop="8dp"
61         app:layout_constraintTop_toBottomOf="@+id/tvLokasi"
62         app:layout_constraintStart_toStartOf="parent"
63         app:layout_constraintEnd_toEndOf="parent" />
64
65     </androidx.constraintlayout.widget.ConstraintLayout>
66 </ScrollView>
```

8. item_gunung.xml

Tabel 13. Source Code item_gunung.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.cardview.widget.CardView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="wrap_content"
8     android:layout_margin="8dp"
9     app:cardElevation="4dp"
10    app:cardCornerRadius="8dp">
11
12    <androidx.constraintlayout.widget.ConstraintLayout
13        android:layout_width="match_parent"
14        android:layout_height="wrap_content"
15        android:padding="25dp">
16
17        <ImageView
18            android:id="@+id/imgGunung"
19            android:layout_width="120dp"
20            android:layout_height="160dp"
21            android:scaleType="centerCrop"
22            tools:src="@drawable/gunung_merbabu"
23            app:layout_constraintStart_toStartOf="parent"
24            app:layout_constraintTop_toTopOf="parent"
25
26            app:layout_constraintEnd_toStartOf="@+id/linearLayoutText"
27            android:layout_marginEnd="12dp"/>
28
29        <LinearLayout
30            android:id="@+id/linearLayoutText"
31            android:orientation="vertical"
```

```
32         android:layout_width="0dp"
33         android:layout_height="wrap_content"
34         app:layout_constraintStart_toEndOf="@+id/imgGunung"
35         app:layout_constraintTop_toTopOf="parent"
36         app:layout_constraintEnd_toEndOf="parent"
37         android:layout_marginTop="8dp"
38         android:layout_weight="1">
39
40     <TextView
41         android:id="@+id/tvGunungName"
42         android:layout_width="wrap_content"
43         android:layout_height="wrap_content"
44         android:textSize="24sp"
45         android:textStyle="bold"
46         android:textColor="@android:color/black"
47         tools:text="Gunung Semeru" />
48
49     <TextView
50         android:id="@+id/tvGunungLokasi"
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:textSize="14sp"
54         android:textColor="#666666"
55         android:textStyle="bold"
56         tools:text="Lokasi: Jawa Timur" />
57
58     <TextView
59         android:id="@+id/tvGunungDeskripsi"
60         android:layout_width="wrap_content"
61         android:layout_height="wrap_content"
62         android:textSize="14sp"
63         android:textColor="@android:color/black"
64         android:maxLines="2"
65         android:ellipsize="end"
```

```
66          tools:text="Gunung tertinggi di Jawa Timur
67 dengan pemandangan yang sangat indah dan memiliki trek yang
68 menantang untuk para pendaki pemula maupun profesional." />
69      </LinearLayout>
70
71      <LinearLayout
72          android:orientation="horizontal"
73          android:layout_width="match_parent"
74          android:layout_height="wrap_content"
75          android:gravity="end"
76          android:layout_marginTop="8dp"
77          android:layout_marginStart="130dp"
78          app:layout_constraintStart_toStartOf="parent"
79          app:layout_constraintEnd_toEndOf="parent"
80
81          app:layout_constraintTop_toBottomOf="@+id/linearLayoutText"
82          android:weightSum="2">
83
84          <Button
85              android:id="@+id/btnLink"
86              android:layout_width="0dp"
87              android:layout_height="wrap_content"
88              android:layout_weight="1"
89              android:text="Link"
90              android:layout_marginEnd="8dp" />
91
92          <Button
93              android:id="@+id/btnDetail"
94              android:layout_width="0dp"
95              android:layout_height="wrap_content"
96              android:layout_weight="1"
97              android:text="Detail" />
98      </LinearLayout>
99
```

```
100  
101  
102     </androidx.constraintlayout.widget.ConstraintLayout>  
103 </androidx.cardview.widget.CardView>
```

9. list_fragment.xml

Tabel 14. Source Code list_fragment.xml

```
1 <?xml version="1.0" encoding="utf-8"?>  
2 <androidx.constraintlayout.widget.ConstraintLayout  
3     xmlns:android="http://schemas.android.com/apk/res/android"  
4     xmlns:app="http://schemas.android.com/apk/res-auto"  
5     xmlns:tools="http://schemas.android.com/tools"  
6     android:layout_width="match_parent"  
7     android:layout_height="match_parent"  
8     tools:context=".ListFragment">  
9  
10    <androidx.recyclerview.widget.RecyclerView  
11        android:id="@+id/rvGunung"  
12        android:layout_width="0dp"  
13        android:layout_height="0dp"  
14        android:clipToPadding="false"  
15        android:background="#94C2EB"  
16        android:padding="16dp"  
17        app:layout_constraintTop_toTopOf="parent"  
18        app:layout_constraintBottom_toBottomOf="parent"  
19        app:layout_constraintStart_toStartOf="parent"  
20        app:layout_constraintEnd_toEndOf="parent"  
21        tools:listitem="@layout/item_gunung" />  
22 </androidx.constraintlayout.widget.ConstraintLayout>  
23
```

app\src\main\res\values

10. colors.xml

Tabel 15. Source Code colors.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="blue_500">#2196F3</color>
4     <color name="blue_700">#1976D2</color>
5 </resources>
6
```

11. string.xml

Tabel 16. Source Code string.xml

```
1 <resources>
2     <string name="app_name">MODUL 3</string>
3     <string-array name="gunung_name">
4         <item>Gunung Bromo</item>
5         <item>Gunung Kerinci</item>
6         <item>Gunung Merapi</item>
7         <item>Gunung Merbabu</item>
8         <item>Gunung Rinjani</item>
9     </string-array>
10
11    <string-array name="gunung_link">
12        <item>https://bromotenggersemeru.org/</item>
13        <item>https://tnkerinciseblat.com/</item>
14        <item>https://tngmerapi.id/</item>
15        <item>https://tngunungmerbabu.org/</item>
16        <item>https://www.rinjanationalpark.id/</item>
17    </string-array>
18
19    <string-array name="gunung_lokasi">
20        <item>Jawa Timur</item>
```

```
21      <item>Jambi, di Taman Nasional Kerinci Seblat</item>
22      <item>Perbatasan Yogyakarta dan Jawa Tengah</item>
23      <item>Jawa Tengah</item>
24      <item>Lombok, Nusa Tenggara Barat</item>
25  </string-array>
26
27  <string-array name="gunung_deskripsi">
28      <item>Gunung Bromo adalah gunung api aktif yang
29 terkenal secara internasional karena lanskapnya yang ikonik
30 dan mudah diakses dengan ketinggian 2.329 mdpl. Gunung ini
31 merupakan bagian dari kaldera Tengger yang sangat luas, dengan
32 lautan pasir (Segara Wedi) selebar sekitar 10 km yang
33 mengelilinginya. Kawah Bromo masih mengeluarkan asap putih dan
34 terkadang belerang, menjadikannya gunung yang terus dipantau
35 meskipun menjadi destinasi wisata utama. Bromo memiliki peran
36 penting dalam budaya masyarakat Suku Tengger, keturunan dari
37 kerajaan Majapahit, yang memegang teguh tradisi Hindu. Salah
38 satu upacara besar mereka adalah Yadnya Kasada, di mana mereka
39 melemparkan hasil panen, ternak, dan sesajen lainnya ke kawah
40 Bromo sebagai persembahan kepada para dewa. Upacara ini
41 menarik banyak wisatawan setiap tahun. Akses menuju Bromo
42 cukup mudah, terutama dari Cemoro Lawang (Probolinggo) dan
43 Wonokitri (Pasuruan). Wisatawan biasanya menantikan momen
44 matahari terbit dari Penanjakan, sebuah titik pandang di sisi
45 timur kaldera yang menawarkan panorama menakjubkan, dengan
46 Gunung Bromo, Batok, dan Semeru berjejer dalam satu garis
47 cakrawala.</item>
48      <item>Gunung Kerinci adalah puncak tertinggi di Pulau
49 Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan
50 total ketinggian 3.805 mdpl. Gunung ini berdiri megah di
51 tengah kawasan konservasi Taman Nasional Kerinci Seblat
52 (TNKS), yang juga merupakan situs Warisan Dunia UNESCO dalam
53 kategori Tropical Rainforest Heritage of Sumatra. Kawasan
54 sekitar Kerinci adalah salah satu titik keanekaragaman hayati
```

55 terkaya di dunia, menjadi rumah bagi harimau Sumatera, tapir,
56 beruang madu, dan banyak jenis burung endemik. Kerinci masih
57 aktif dan kerap menunjukkan aktivitas vulkanik berupa letusan
58 kecil, gempa vulkanik, dan hembusan asap kawah. Meskipun
59 begitu, gunung ini tetap menjadi daya tarik pendakian bagi
60 pencinta alam ekstrem. Jalur utama pendakian melalui desa
61 Kersik Tuo memiliki trek yang panjang dan menantang, melewati
62 hutan hujan tropis lebat, rawa, dan jalur batu curam. Dari
63 puncaknya, pendaki dapat melihat garis pantai Samudera Hindia
64 dan Pegunungan Bukit Barisan yang menyapu cakrawala. Selain
65 sebagai objek wisata, Kerinci juga penting untuk penelitian
66 geologi dan pelestarian ekosistem pegunungan tropis.</item>
67 <item>Gunung Merapi merupakan gunung berapi paling
68 aktif di Indonesia dan salah satu yang paling aktif di dunia
69 dengan total ketinggian 2.930 mdpl. Merapi secara rutin
70 mengalami letusan setiap 2–5 tahun sekali dan sangat
71 memengaruhi wilayah padat penduduk di sekitarnya. Letusan
72 besar terakhir yang menyebabkan korban jiwa terjadi pada tahun
73 2010, menewaskan puluhan orang dan memaksa ribuan lainnya
74 mengungsi. Merapi bukan sekadar gunung, tetapi juga simbol
75 budaya dan spiritualitas masyarakat Jawa. Gunung ini dipercaya
76 sebagai pusat dunia spiritual dalam kosmologi Keraton
77 Yogyakarta. Setiap tahun, masyarakat melakukan ritual Labuhan
78 Merapi sebagai bentuk persembahan dan penghormatan terhadap
79 kekuatan alam. Selain itu, kawasan lereng Merapi menjadi objek
80 wisata edukasi, seperti Museum Gunungapi Merapi dan tur lava
81 jeep yang memperlihatkan bekas jalur aliran awan panas (wedhus
82 gembel). Secara geologis, Merapi terus dimonitor secara
83 intensif oleh PVMBG dan BPPTKG dengan berbagai alat modern
84 seperti seismograf, kamera thermal, dan satelit. Pendaki
85 umumnya hanya diperbolehkan naik hingga Pasar Bubrah, karena
86 puncaknya sangat berisiko terkena guguran lava dan awan
87 panas..</item>
88 <item>Gunung Merbabu adalah gunung dengan ketinggian

89 3.145 meter di atas permukaan laut (mdpl) yang bertipe
90 stratovolcano yang sudah tidak aktif, berdampingan erat dengan
91 Gunung Merapi di sebelah selatannya. Nama "Merbabu" berasal
92 dari gabungan kata "Meru" (gunung) dan "Abu", yang secara
93 harfiah berarti "gunung abu". Gunung ini memiliki keunikan
94 berupa hamparan padang sabana luas yang sangat memikat,
95 terutama saat musim kemarau ketika rerumputan menguning
96 keemasan. Pendaki umumnya memilih jalur via Selo (Boyolali)
97 atau Wekas (Magelang) karena pemandangan spektakuler dan trek
98 yang relatif ramah. Gunung Merbabu sangat populer di kalangan
99 pendaki, terutama karena spot sunrise dari puncaknya yang
100 memperlihatkan lanskap gunung lain seperti Merapi, Sumbing,
101 Sindoro, Lawu, dan bahkan Slamet. Flora dan fauna di kawasan
102 ini cukup beragam, termasuk edelweiss, burung jalak, serta
103 lutung Jawa. Meskipun secara vulkanik tidak aktif, Merbabu
104 tetap diawasi karena posisinya yang dekat dengan Merapi yang
105 sangat aktif. Selain pendakian, wilayah di kaki Merbabu juga
106 dimanfaatkan untuk pertanian hortikultura oleh warga
107 lokal.</item>
108 <item>Gunung Rinjani merupakan gunung berapi tertinggi
109 kedua di Indonesia dan menjadi ikon Pulau Lombok dengan
110 ketinggian 3.726 mdpl. Gunung ini termasuk dalam Taman
111 Nasional Gunung Rinjani yang luasnya mencapai lebih dari
112 41.000 hektare. Di kawahnya terdapat Danau Segara Anak, yang
113 menjadi pusat spiritual dan simbol kehidupan bagi masyarakat
114 lokal. Di tengah danau tersebut, menjulang Gunung Barujari,
115 yang merupakan kawah aktif dari Rinjani dan menjadi sumber
116 letusan terakhir pada 2016. Pendakian Rinjani dikenal berat
117 dan membutuhkan stamina serta persiapan fisik yang matang.
118 Jalur-jalur populer antara lain via Sembalun (lebih terbuka
119 dan panas) serta Senaru (lebih rimbun dan teduh). Rinjani
120 menawarkan pemandangan yang menakjubkan: dari hutan tropis,
121 air terjun, danau, hingga puncak berbatu yang curam. Selain
122 nilai geologis, Rinjani juga memiliki arti spiritual dan

```

123 budaya tinggi, seperti ritual Pekelan atau Mulang Pakelem yang
124 dilakukan masyarakat Bali dan Sasak di Danau Segara Anak
125 sebagai bentuk penghormatan kepada alam.
126
127 </item>
128     </string-array>
129
130     <array name="gunung_image">
131         <item>@drawable/gunung_bromo</item>
132         <item>@drawable/gunung_kerinci</item>
133         <item>@drawable/gunung_merapi</item>
134         <item>@drawable/gunung_merbabu</item>
135         <item>@drawable/gunung_rinjani</item>
136     </array>
137 </resources>

```

12. theme.xml

Tabel 17. Source Code theme.xml

```

1 <resources xmlns:tools="http://schemas.android.com/tools">
2     <!-- Base application theme. -->
3     <style name="Base.Theme.Modul3"
4 parent="Theme.Material3.DayNight.NoActionBar">
5         <!-- Tambahkan warna utama -->
6         <item name="colorPrimary">@color/blue_500</item>
7         <item name="colorOnPrimary">@android:color/white</item>
8         <item
9             name="colorPrimaryContainer">@color/blue_700</item>
10        <item
11            name="colorOnPrimaryContainer">@android:color/white</item>
12        </style>
13
14     <style name="Theme.Modul3" parent="Base.Theme.Modul3" />
15 </resources>

```

app\src\main

13. AndroidManifest.xml

Table 18. Source Code AndroidManifest.xml

```
1 package com.example.modul3
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.modul3.databinding.ItemGunungBinding
7
8 class GunungAdapter(
9     private val listGunung: ArrayList<Gunung>,
10    private val onLinkClick: (String) -> Unit,
11    private val onDetailClick: (Int, String, String, String) ->
12 Unit
13 ) : RecyclerView.Adapter<GunungAdapter.ListViewHolder>() {
14
15     class ListViewHolder(private val binding:
16 ItemGunungBinding) : RecyclerView.ViewHolder(binding.root) {
17         fun bind(gunung: Gunung, onLinkClick: (String) -> Unit,
18 onDetailClick: (Int, String, String, String) -> Unit) {
19             // Bind data to views using the binding object
20             binding.tvGunungName.text = gunung.name
21             binding.tvGunungLokasi.text = gunung.lokasi
22             binding.tvGunungDeskripsi.text = gunung.deskripsi
23             binding.imgGunung.setImageResource(gunung.image)
24
25             binding.btnLink.setOnClickListener {
26                 onLinkClick(gunung.link) }
27
28             binding.btnDetail.setOnClickListener {
29                 onDetailClick(gunung.image, gunung.name,
30 gunung.lokasi, gunung.deskripsi)
```

```

31         }
32     }
33 }
34
35     override fun onCreateViewHolder(parent: ViewGroup,
36 viewType: Int): ListViewHolder {
37         val binding =
38             ItemGunungBinding.inflate(LayoutInflater.from(parent.context),
39             parent, false)
40         return ListViewHolder(binding)
41     }
42
43     override fun getItemCount(): Int = listGunung.size
44
45     override fun onBindViewHolder(holder: ListViewHolder,
46 position: Int) {
47         val gunung = listGunung[position]
48         holder.bind(gunung, onLinkClick, onDetailClick)
49     }
50 }
```

14. build.gradle.kts

Tabel 19. Source Code BuildGradle.kts

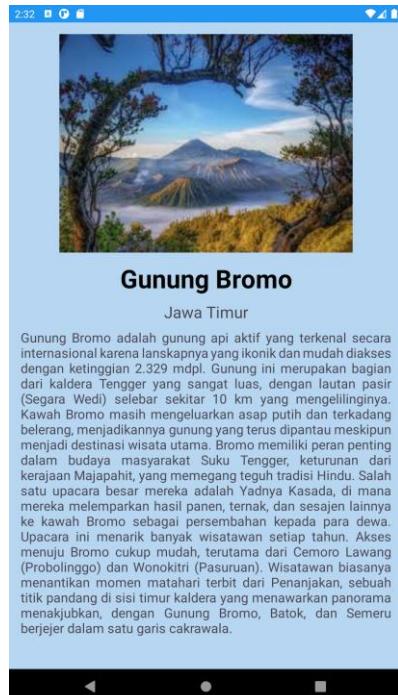
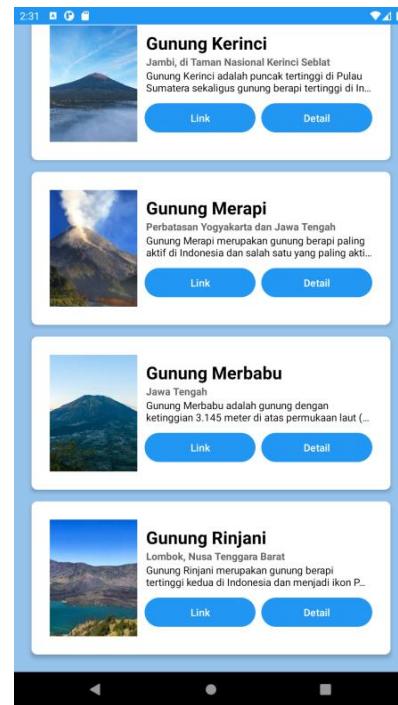
```

1 plugins {
2     id("com.android.application")
3     id("org.jetbrains.kotlin.android")
4     id("kotlin-parcelize")
5 }
6
7 android {
8     namespace = "com.example.modul3"
9     compileSdk = 34
10 }
```

```
11     defaultConfig {
12         applicationId = "com.example.modul3"
13         minSdk = 30
14         targetSdk = 34
15         versionCode = 1
16         versionName = "1.0"
17
18         testInstrumentationRunner =
19 "androidx.test.runner.AndroidJUnitRunner"
20     }
21
22     buildTypes {
23         release {
24             isMinifyEnabled = false
25             proguardFiles(
26                 getDefaultProguardFile("proguard-android-
27 optimize.txt"),
28                 "proguard-rules.pro"
29             )
30         }
31     }
32
33     compileOptions {
34         sourceCompatibility = JavaVersion.VERSION_17
35         targetCompatibility = JavaVersion.VERSION_17
36     }
37
38     kotlinOptions {
39         jvmTarget = "17"
40     }
41
42     buildFeatures {
43         viewBinding = true
44     }
```

```
45 }
46
47 dependencies {
48     implementation("androidx.core:core-ktx:1.12.0")
49     implementation("org.jetbrains.kotlin:kotlin-stdlib:1.9.0")
50     implementation("androidx.appcompat:appcompat:1.6.1")
51     implementation("com.google.android.material:material:1.11.0")
52
53     implementation("androidx.constraintlayout:constraintlayout:2.1.4")
54     implementation("androidx.recyclerview:recyclerview:1.3.2")
55     implementation("androidx.cardview:cardview:1.0.0")
56     implementation("androidx.lifecycle:lifecycle-runtime-
57 ktx:2.6.2")
58     implementation("androidx.activity:activity-ktx:1.8.2")
59
60     testImplementation("junit:junit:4.13.2")
61     androidTestImplementation("androidx.test.ext:junit:1.1.5")
62     androidTestImplementation("androidx.test.espresso:espresso-
63 core:3.5.1")
64 }
```

B. Output Program



2:34

Gunung Bromo
Jawa Timur
Gunung Bromo adalah gunung api aktif yang terkenal secara internasional karena lanskapnya yang ikonik dan mudah diakses dengan ketinggian 2.329 mdpl. Gunung ini merupakan bagian dari kaldera Tengger yang ...

[Link](#) [Detail](#)

Gunung Kerinci
Jambi, di Taman Nasional Kerinci Seblat
Gunung Kerinci adalah puncak tertinggi di Pulau Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan total ketinggian 3.805 mdpl. Gunung ini berdiri megah di tengah kawasan konservasi Taman Nasional...

[Link](#) [Detail](#)

2:34

Gunung Merbabu
Jawa Tengah
Gunung Merbabu adalah gunung dengan ketinggian 3.145 meter di atas permukaan laut (mdpl) yang bertipe stratovolcano yang sudah tidak aktif, berdampingan erat dengan Gunung Merapi di sebelah selatannya. Nam...

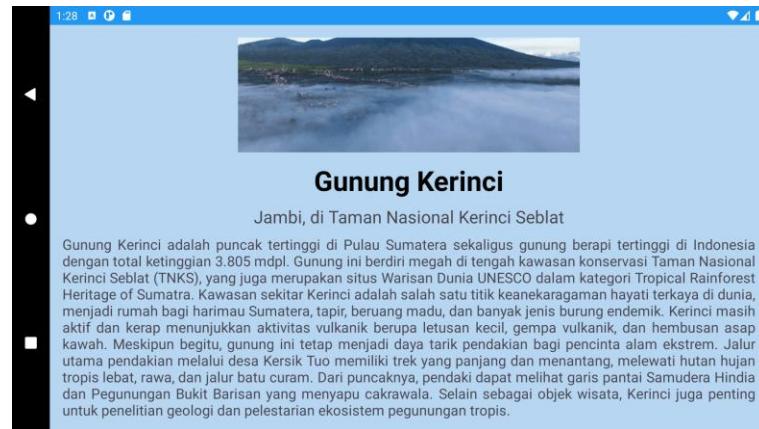
[Link](#) [Detail](#)

Gunung Rinjani
Lombok, Nusa Tenggara Barat
Gunung Rinjani merupakan gunung berapi tertinggi kedua di Indonesia dan menjadi ikon Pulau Lombok dengan ketinggian 3.726 mdpl. Gunung ini termasuk dalam Taman Nasional Gunung Rinjani yang luasnya me...

[Link](#) [Detail](#)

2:35

Gunung Kerinci
Jambi, di Taman Nasional Kerinci Seblat
Gunung Kerinci adalah puncak tertinggi di Pulau Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan total ketinggian 3.805 mdpl. Gunung ini berdiri megah di tengah kawasan konservasi Taman Nasional Kerinci Seblat (TNKS), yang juga merupakan situs Warisan Dunia UNESCO dalam kategori Tropical Rainforest Heritage of Sumatra. Kawasan sekitar Kerinci adalah salah satu titik keanekaragaman hayati terkaya di dunia



Gambar 3. Screenshot Hasil Jawaban Soal 1Modul 3

C. Pembahasan

app\src\main\java\com\example\modul3

1. DetailFragment.kt

Pada baris [1] terdapat fungsi package com.example.modul3 yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.modul3. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [8] terdapat fungsi import yang digunakan untuk mengimpor berbagai komponen yang dibutuhkan dalam pengembangan aplikasi Android, diantaranya untuk menampilkan detail informasi berupa nama, lokasi, deskripsi, dan gambar, yang dikirim melalui Bundle arguments, ViewBinding (DetailFragmentBinding) untuk mengakses view secara aman, dan mengatur ulang binding saat view dihancurkan guna mencegah memory leak. Pada baris [10] digunakan untuk mendeklarasikan sebuah kelas bernama DetailFragment yang merupakan subclass dari Fragment dalam Android. Pada baris [12] dan [13] terdapat fungsi private var _binding: DetailFragmentBinding? = null dan private val binding get() = _binding!! digunakan untuk mengelola ViewBinding dalam sebuah fragment secara aman. Variabel _binding bertipe nullable (DetailFragmentBinding?) berfungsi untuk menyimpan objek binding yang menghubungkan class Kotlin dengan layout XML (detail_fragment.xml). Binding ini diinisialisasi saat onCreateView() dan dihapus

(null) pada `onDestroyView()` guna mencegah kebocoran memori karena siklus hidup View pada fragment bisa berbeda dari fragment itu sendiri. Sementara itu, properti binding merupakan versi non-null dari `_binding`, yang memanfaatkan operator `!!` untuk memastikan bahwa binding hanya digunakan ketika sudah pasti tidak null. Dengan pendekatan ini, kita dapat mengakses elemen-elemen view secara aman dan efisien tanpa perlu memanggil `findViewById`, serta tetap menjaga praktik pemrograman yang sesuai dengan lifecycle fragment. Pada baris [15] hingga [20] terdapat fungsi yang digunakan untuk membuat dan mengembalikan tampilan (view) dari fragment saat fragment sedang dibuat. Parameter inflater digunakan untuk "meng-inflate" layout XML menjadi objek view, sedangkan container adalah parent view tempat fragment akan ditempelkan, dan `savedInstanceState` menyimpan data keadaan sebelumnya jika ada. Di dalam metode ini, digunakan `DetailFragmentBinding.inflate(...)` untuk menghubungkan layout `detail_fragment.xml` dengan objek binding `_binding`. Proses inflate ini membuat layout XML bisa diakses melalui properti binding, sehingga memudahkan dalam pengelolaan tampilan secara efisien tanpa perlu `findViewById`. Setelah di-inflate, biasanya method ini akan mengembalikan `binding.root` sebagai tampilan utama fragment (meskipun belum terlihat di sini). Pendekatan ini memastikan integrasi yang aman antara tampilan dan logika dalam fragment, serta mengikuti praktik modern pengembangan Android menggunakan ViewBinding. Pada baris [22] hingga [25] terdapat fungsi kode yang digunakan untuk **mengambil data yang dikirimkan ke fragment melalui Bundle arguments**. Masing-masing baris berfungsi untuk mengambil data berdasarkan key yang telah ditentukan, seperti `"EXTRA_PHOTO"` untuk gambar (tipe Int), `"EXTRA_NAME"` untuk nama (tipe String), `"EXTRA_LOKASI"` untuk lokasi, dan `"EXTRA_DESKRIPSI"` untuk deskripsi. Metode `getInt()` dan `getString()` dipanggil secara aman menggunakan operator `?,`, yang artinya data hanya akan diambil jika arguments tidak bernilai null. Biasanya, data ini dikirim dari fragment atau activity sebelumnya saat navigasi ke DetailFragment, dan nantinya akan ditampilkan ke dalam elemen UI fragment. Pada baris [28] hingga [25] terdapat fungsi kode yang menampilkan data yang diterima dari arguments ke dalam elemen-elemen UI pada fragment menggunakan

ViewBinding. Pertama, nilai variabel name, lokasi, dan deskripsi yang telah diambil dari arguments diset ke dalam TextView yang sesuai, seperti tvName, tvLokasi, dan tvDeskripsi. Hal ini memungkinkan fragment untuk menampilkan informasi seperti nama, lokasi, dan deskripsi pada tampilan UI. Selanjutnya, jika ada nilai untuk gambar (yang disimpan dalam variabel image), kode menggunakan let untuk memastikan bahwa nilai tersebut tidak null sebelum mengatur gambar pada ImageView (imgPoster) dengan menggunakan setImageResource(). Terakhir, metode return binding.root mengembalikan root view dari layout yang telah di-bind, yaitu tampilan fragment yang sudah lengkap dengan data yang ditampilkan. Pendekatan ini menggabungkan ViewBinding untuk mengakses tampilan UI dengan cara yang lebih aman dan efisien, menghindari penggunaan findViewById yang rentan terhadap kesalahan dan meningkatkan keterbacaan kode. Pada baris [38] hingga [40] terdapat Metode onDestroyView(), bagian dari siklus hidup fragment yang dipanggil ketika tampilan fragment akan dihancurkan atau saat fragment tidak lagi ditampilkan di layar. Dalam metode ini, super.onDestroyView() dipanggil untuk memastikan bahwa proses penghancuran tampilan fragment dilakukan dengan benar oleh sistem. Setelah itu, _binding = null digunakan untuk **menghapus referensi ke objek binding** dan mencegah terjadinya **memory leak**. Hal ini penting karena meskipun fragment dihancurkan, tampilan (view) yang digunakan oleh fragment bisa tetap ada di memori jika referensinya tidak dihapus. Dengan menyetel _binding ke null, kita memastikan bahwa objek tersebut tidak lagi mengacu pada tampilan yang sudah dihancurkan, sehingga sumber daya dapat dibebaskan dan memori tidak terbuang sia-sia. Pendekatan ini adalah best practice untuk mengelola siklus hidup fragment dan mencegah masalah terkait memori dalam aplikasi Android.

2. Gunung.kt

Pada baris [1] tersebut mendefinisikan sebuah kelas data bernama Gunung yang mengimplementasikan interface Parcelable. Pada baris [7] hingga [12] menampilkan lima property yakni image (menyimpan ID sumber daya gambar), name (nama gunung), lokasi (lokasi gunung), deskripsi (deskripsi gunung), dan link (URL terkait gunung). Dengan menambahkan anotasi @Parcelize, kelas Gunung

secara otomatis mendapatkan implementasi dari metode Parcelable, yang memungkinkan objek dari kelas ini untuk diserialisasi dan dipassing antar komponen dalam aplikasi Android, seperti Activity atau Fragment, melalui Intent atau Bundle. Anotasi ini menyederhanakan proses karena Android akan menangani semua detail terkait serialisasi dan deserialisasi objek, sehingga developer tidak perlu menulis kode secara manual. Keuntungan utama dari menggunakan Parcelable adalah efisiensi kinerja dalam mengirim data antar komponen di aplikasi, yang lebih cepat dibandingkan menggunakan Serializable. Dengan pendekatan ini, objek Gunung dapat dengan mudah dipindahkan dan dipertukarkan antar aktivitas atau fragment dalam aplikasi.

3. GunungAdapter.kt

Pada baris [1] terdapat fungsi package com.example.modul3 yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.modul3. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [6] terdapat beberapa import yakni mengimpor LayoutInflater, ViewGroup, dan RecyclerView untuk membuat dan mengelola daftar item dengan menggunakan RecyclerView di Android. ItemGunungBinding digunakan untuk menghubungkan elemen-elemen dalam layout item_gunung.xml dengan kode, memungkinkan akses langsung ke elemen UI tanpa menggunakan findViewById. Ini mempermudah pengelolaan tampilan dalam setiap item daftar di RecyclerView. Pada baris [8] hingga [13] terdapat kelas GunungAdapter, adapter untuk RecyclerView yang menghubungkan data Gunung dengan tampilan item dalam daftar. Kelas ini menerima tiga parameter: listGunung (daftar objek Gunung), onLinkClick (fungsi lambda untuk menangani klik pada link), dan onDetailClick (fungsi lambda untuk menangani klik pada item untuk menampilkan detail). Dengan menggunakan RecyclerView.Adapter, kelas ini mengelola tampilan daftar secara efisien dan memungkinkan interaksi pengguna dengan item, seperti membuka link atau melihat detail gunung. Pada baris [15] hingga [30] terdapat kelas ListViewHolder berfungsi untuk mengikat data dari objek Gunung ke tampilan item di RecyclerView. Dalam

metode bind(), data dari objek Gunung (seperti nama, lokasi, deskripsi, dan gambar) di-set ke elemen-elemen tampilan melalui objek binding. Selain itu, dua tombol aksi, yaitu btnLink dan btnDetail, di-set dengan listener klik yang memanggil fungsi lambda onLinkClick dan onDetailClick dengan parameter terkait, seperti link dan detail gunung. Pendekatan ini memastikan bahwa setiap item dalam daftar dapat diinteraksikan dengan mudah dan menampilkan data dengan benar. Pada baris [35] hingga [48] terdapat metode onCreateViewHolder() digunakan untuk membuat dan mengembalikan objek ListViewHolder dengan meng-inflate layout ItemGunungBinding ke dalam tampilan item. onBindViewHolder() akan mengikat data dari objek Gunung yang ada pada posisi tertentu di listGunung ke dalam ListViewHolder, dengan memanggil metode bind() untuk mengisi tampilan dengan data yang sesuai dan mengatur aksi klik pada tombol. Sedangkan getItemCount() mengembalikan jumlah item dalam daftar listGunung, yang menentukan banyaknya item yang akan ditampilkan di RecyclerView. Kode ini memastikan setiap item di RecyclerView terikat dengan data yang benar dan interaktif.

4. ListFragment.kt

Pada baris [1] terdapat fungsi package com.example.modul3 yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.modul3. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [11] terdapat beberapa import yakni mengimpor komponen untuk mengelola tampilan Fragment yang menampilkan daftar dengan RecyclerView. Intent dan Uri digunakan untuk membuka link eksternal, sedangkan LinearLayoutManager menyusun item secara vertikal. ListFragmentBinding menghubungkan layout XML dengan kode untuk mempermudah pengelolaan elemen UI. Pada baris [13] hingga [19] terdapat kelas ListFragment, adalah Fragment yang mengelola tampilan daftar menggunakan RecyclerView. Di dalamnya, terdapat dua properti penting: _binding yang menghubungkan layout dengan kode menggunakan ListFragmentBinding, dan binding untuk mengakses elemen UI dengan aman. Selain itu, terdapat properti gunungAdapter, yang merupakan adapter untuk menampilkan data Gunung dalam RecyclerView, serta list yang berfungsi menyimpan data Gunung dalam bentuk

ArrayList. Kode ini mempersiapkan Fragment untuk menampilkan daftar gunung dengan data yang akan diisi kemudian. Pada baris [21] hingga [32] terdapat metode onCreateView meng-inflate layout ListFragmentBinding ke dalam tampilan fragment menggunakan inflater. Kemudian, daftar list dibersihkan dengan list.clear() dan diisi dengan data dari fungsi getListGunung(). Setelah itu, metode setupRecyclerView() dipanggil untuk menyiapkan dan mengatur RecyclerView agar dapat menampilkan data gunung dalam daftar. Terakhir, metode ini mengembalikan binding.root sebagai tampilan utama dari fragment, yang akan ditampilkan kepada pengguna. Pada baris [35] hingga [49] terdapat metode setupRecyclerView menginisialisasi gunungAdapter dengan daftar list yang berisi data gunung. Dua fungsi lambda diteruskan sebagai parameter: onLinkClick, yang membuka URL menggunakan Intent, dan onDetailClick, yang memulai DetailFragment dengan membawa data detail gunung (seperti gambar, nama, lokasi, dan deskripsi) melalui Bundle. Aksi klik pada item di daftar akan menampilkan fragment detail atau membuka link eksternal. Dengan demikian, metode ini memastikan interaksi pengguna berjalan sesuai yang diinginkan pada RecyclerView. Pada baris [52] hingga [56] terdapat transaksi fragment dengan menggunakan parentFragmentManager.beginTransaction() digunakan untuk memulai transaksi, kemudian replace(R.id.frame_container, detailFragment) menggantikan tampilan yang ada di dalam container (dengan ID frame_container) dengan detailFragment. Metode addToBackStack(null) memastikan bahwa transaksi fragment ini ditambahkan ke stack kembali (back stack), sehingga pengguna bisa kembali ke fragment sebelumnya. Terakhir, commit() menjalankan transaksi dan mengganti tampilan di dalam fragment. Pada baris [60] hingga [63] berguna untuk mengonfigurasi RecyclerView yang diakses melalui binding.rvGunung. Pertama, layoutManager diatur ke LinearLayoutManager(context), yang menyusun item dalam daftar secara vertikal. Kemudian, adapter diatur ke gunungAdapter, yang bertanggung jawab untuk menyediakan data ke RecyclerView. Terakhir, setHasFixedSize(true) memberi tahu RecyclerView bahwa ukuran tampilan item tidak akan berubah, yang dapat meningkatkan performa saat meng gulir. Pada baris [67] hingga [77] terdapat fungsi getListGunung mengambil data terkait gunung dari sumber daya aplikasi.

`resources.obtainTypedArray(R.array.gunung_image)` digunakan untuk mendapatkan array berisi referensi gambar gunung, sementara `resources.getStringArray()` digunakan untuk mengambil array data nama gunung, lokasi, deskripsi, dan link terkait. Data-data ini kemudian digunakan untuk membuat objek Gunung yang disimpan dalam `ArrayList<Gunung>` dan akhirnya dikembalikan sebagai hasil fungsi. Pada baris [80] hingga [87] terdapat fungsi untuk membuat objek `ArrayList<Gunung>` bernama `listGunung`, lalu mengisi daftar tersebut dengan data dari array sumber daya. Melalui perulangan `for`, setiap elemen array digunakan untuk membuat objek Gunung dengan memasukkan gambar (dari `dataImage.getResourceId()`), nama, lokasi, deskripsi, dan link sesuai indeks. Objek Gunung yang telah dibuat ditambahkan ke dalam list. Setelah selesai, `dataImage.recycle()` dipanggil untuk membebaskan memori yang digunakan oleh array gambar. Fungsi kemudian mengembalikan `listGunung` yang sudah berisi data lengkap. Pada baris [90] hingga [92] terdapat metode `nDestroyView()` dipanggil saat tampilan fragment dihancurkan, biasanya ketika fragment tidak lagi terlihat di layar. Di dalamnya, `super.onDestroyView()` dipanggil terlebih dahulu untuk menjalankan logika bawaan Android. Kemudian `_binding` diset ke `null` untuk memutus referensi view binding yang sebelumnya digunakan. Ini adalah praktik umum dalam penggunaan ViewBinding di fragment untuk mencegah kebocoran memori, karena objek binding masih menyimpan referensi ke tampilan meskipun tampilan tersebut sudah tidak digunakan.

5. MainActivity.kt

Pada baris [1] terdapat fungsi `package com.example.modul3` yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama `com.example.modul3`. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] dan [4] terdapat beberapa import, baris import `android.os.Bundle` digunakan untuk mengimpor kelas Bundle, yaitu objek yang berfungsi menyimpan dan mengelola data sementara yang digunakan dalam pengiriman data antar komponen Android, seperti saat menyimpan state activity. Sedangkan import

androidx.appcompat.app.AppCompatActivity digunakan untuk mengimpor AppCompatActivity, yaitu kelas dasar untuk activity yang memberikan dukungan kompatibilitas ke versi Android lama dan memungkinkan penggunaan fitur modern seperti Toolbar dan Fragment secara lebih konsisten. Keduanya biasanya digunakan saat membuat kelas Activity di Android. Pada baris [7] hingga [15] terdapat fungsi kelas MainActivity mewarisi AppCompatActivity, dan di dalam metode onCreate() dilakukan inisialisasi tampilan utama menggunakan setContentView(R.layout.activity_main). Selanjutnya, supportFragmentManager digunakan untuk mengelola fragment. Sebuah instance ListFragment dibuat dan disiapkan untuk ditambahkan ke MainActivity. Kemudian, baris findFragmentByTag(...) digunakan untuk memeriksa apakah fragment dengan tag nama kelas ListFragment sudah ada sebelumnya, yang berguna untuk menghindari penambahan fragment secara berulang saat konfigurasi ulang seperti rotasi layar. Pada baris [17] hingga [22] terdapat fungsi yang berguna untuk memeriksa apakah fragment dengan tag ListFragment belum ada pada fragmentManager. Jika belum (fragment != ListFragment), maka dilakukan transaksi fragment menggunakan beginTransaction(). Melalui add(), fragment listFragment ditambahkan ke dalam R.id.frame_container, yaitu sebuah ViewGroup di layout activity_main.xml yang menjadi wadah tampilan fragment. Tag yang digunakan adalah nama kelas ListFragment agar fragment ini bisa dikenali kembali di kemudian waktu. Akhirnya, commit() digunakan untuk menyelesaikan dan mengeksekusi transaksi fragment tersebut secara efektif. Pendekatan ini umum dalam aplikasi Android berbasis fragment untuk mengatur tampilan secara modular.

app\src\main\res\layout

6. **activity_main.xml**

Kode pada kelas ini merupakan layout activity_main.xml yang digunakan sebagai tampilan utama dari MainActivity dalam aplikasi Android. Layout ini menggunakan FrameLayout sebagai elemen root, yang berfungsi sebagai wadah (container) untuk menampung satu atau beberapa tampilan (biasanya Fragment) di dalamnya. Atribut android:id="@+id/frame_container" memberikan ID pada

FrameLayout, sehingga dapat diakses dan dimanipulasi dari kode Kotlin, misalnya saat menambahkan ListFragment ke dalamnya. Ukuran layout diatur memenuhi layar dengan match_parent untuk lebar dan tinggi. Namespace tools:context menunjukkan bahwa layout ini akan digunakan oleh MainActivity, dan digunakan oleh Android Studio untuk tujuan preview. FrameLayout dipilih karena cocok untuk menampilkan satu Fragment pada satu waktu secara bertumpuk.

7. **detail_fragment.xml**

Kode xml ini merupakan layout untuk detail_fragment.xml yang digunakan oleh DetailFragment.kt. Pada baris [1] hingga [10] terdapat layout yang menggunakan elemen ScrollView sebagai root, yang memungkinkan konten di dalamnya untuk dapat digulir secara vertikal jika melebihi tinggi layar. Atribut layout_width dan layout_height disetel ke match_parent, sehingga ScrollView akan mengisi seluruh layar. Warna latar belakang diatur ke #B6D6F1, memberikan nuansa biru muda yang lembut. Padding sebesar 16dp ditambahkan ke seluruh sisi untuk memberikan ruang antara konten dan tepi layar. Namespace tools:context menunjukkan bahwa layout ini milik DetailFragment, berguna untuk preview di Android Studio. Umumnya, di dalam ScrollView akan ada LinearLayout vertikal yang berisi elemen-elemen UI seperti gambar, teks, dan tombol. Pada baris [12] hingga [14] terdapat fungsi yang mendefinisikan sebuah ConstraintLayout yang merupakan wadah tata letak fleksibel dalam Android. ConstraintLayout digunakan di dalam ScrollView untuk menyusun elemen-elemen UI secara fleksibel dengan menggunakan constraint atau batasan antar elemen, tanpa harus membuat hierarki layout yang dalam. Atribut layout_width="match_parent" berarti layout ini akan mengambil seluruh lebar dari parent-nya (dalam hal ini, ScrollView), sedangkan layout_height="wrap_content" berarti tinggi layout akan mengikuti tinggi konten di dalamnya. ConstraintLayout sering dipilih karena efisien dalam kinerja dan memungkinkan desain yang kompleks tanpa banyak nesting layout. Di dalamnya biasanya terdapat berbagai elemen UI (seperti ImageView, TextView, Button) yang diatur posisinya relatif terhadap satu sama lain. Pada baris [16] hingga [23] terdapat baris komponen ImageView di dalam ConstraintLayout yang digunakan untuk

menampilkan gambar (dalam konteks ini, kemungkinan gambar gunung). android:id="@+id/imgPoster" memberikan ID unik agar dapat diakses di Kotlin melalui ViewBinding atau findViewById. Atribut layout_width="0dp" digunakan bersama constraint Start dan End, yang berarti lebarnya akan disesuaikan dengan lebar antara batas kiri dan kanan parent-nya. layout_height="300dp" menetapkan tinggi tetap sebesar 300dp. Atribut contentDescription="Foto Gunung" penting untuk aksesibilitas, memberi tahu pengguna pembaca layar bahwa ini adalah gambar gunung. Constraint Top_toTopOf="parent", Start_toStartOf="parent", dan End_toEndOf="parent" memastikan gambar berada di bagian atas dan terpusat secara horizontal dalam layout. Pendekatan ini umum digunakan dalam desain modern Android agar UI responsif dan konsisten di berbagai ukuran layar. Pada baris [25] hingga [37] terdapat fungsi TextView di atas digunakan untuk menampilkan nama gunung dengan tampilan mencolok di bawah gambar. ID @+id/tvName memungkinkan TextView ini diakses melalui kode Kotlin (misalnya dengan ViewBinding). Lebarnya diset 0dp karena menggunakan constraint horizontal (Start dan End) agar mengisi ruang dari kiri ke kanan layout induk. layout_height="wrap_content" membuat tingginya menyesuaikan isi teks. Atribut gravity="center" memusatkan teks dalam TextView, sedangkan textSize="35sp" dan textStyle="bold" membuat teks terlihat besar dan tebal, ideal untuk judul. Warna teks hitam ditentukan dengan textColor="@android:color/black" agar kontras. tools:text="Gunung Semeru" hanya digunakan untuk preview di Android Studio dan tidak muncul saat runtime. Constraint Top_toBottomOf="@id/imgPoster" menempatkan TextView tepat di bawah gambar, sedangkan Start dan End dikaitkan ke parent untuk membuatnya berada di tengah secara horizontal. Layout ini mendukung tampilan yang bersih, responsif, dan estetis. Pada baris [39] hingga [49] terdapat fungsi TextView ini digunakan untuk menampilkan lokasi gunung dan ditempatkan tepat di bawah nama gunung. Lebarnya diatur 0dp agar mengisi ruang horizontal antara batas kiri dan kanan parent layout (menggunakan constraintStart dan constraintEnd). Tingginya otomatis menyesuaikan isi teks karena wrap_content. Teks ditampilkan di tengah menggunakan gravity="center" dan ukuran font-nya 22sp, cocok untuk informasi tambahan. Atribut tools:text="Lokasi: Jawa Timur"

hanya digunakan sebagai contoh pratinjau di Android Studio, bukan untuk tampilan saat aplikasi dijalankan. `layout_marginTop="8dp"` memberi jarak ke atas agar tidak terlalu rapat dengan teks nama gunung. `Constraint Top_toBottomOf="@id/tvName"` menempatkan `TextView` ini tepat di bawah elemen nama, menjaga susunan konten yang rapi dan konsisten. Pada baris [51] hingga [66] terdapat fungsi xml yang digunakan untuk tampilan halaman detail untuk informasi gunung dalam aplikasi Android. Struktur utamanya dibungkus oleh `ScrollView`, yang memungkinkan seluruh konten bisa digulir secara vertikal jika melebihi tinggi layar. Di dalam `ScrollView`, terdapat `ConstraintLayout` yang berfungsi sebagai wadah fleksibel untuk menyusun komponen UI dengan posisi yang saling terikat. Komponen yang digunakan terdiri dari `ImageView` untuk menampilkan gambar gunung di bagian atas, kemudian diikuti oleh tiga `TextView` yang masing-masing menampilkan nama gunung, lokasi, dan deskripsi secara terstruktur. Nama gunung ditampilkan dengan ukuran huruf besar dan tebal agar menjadi fokus utama, sedangkan lokasi ditampilkan dengan ukuran sedang. Deskripsi ditata menggunakan mode perataan antar kata (`justificationMode="inter_word"`) agar terlihat lebih rapi dan nyaman dibaca. Seluruh elemen disusun secara responsif dan simetris di tengah layar, menciptakan antarmuka yang bersih dan informatif bagi pengguna.

8. Item_gunung.xml

Kode xml ini merupakan awal dari layout item yang menggunakan `CardView` sebagai kontainer utama. Pada baris [1] hingga [10] terdapat fungsi `CardView` ini berfungsi untuk membungkus satu item data (misalnya data gunung) dengan tampilan yang rapi dan memiliki efek elevasi (bayangan) serta sudut melengkung. Properti `layout_width` diset ke `match_parent` agar lebar kartu menyesuaikan dengan lebar parent, sedangkan `layout_height` diset ke `wrap_content` agar menyesuaikan tinggi konten di dalamnya. Margin luar diberikan sebesar 8dp agar antar item tidak saling menempel. Efek bayangan diatur melalui `cardElevation` sebesar 4dp dan `cardCornerRadius` sebesar 8dp untuk memberi kesan visual modern dan menarik. Biasanya, elemen UI seperti gambar dan teks akan ditempatkan di dalam `CardView` ini untuk menampilkan informasi setiap item dalam `RecyclerView`. Pada baris [12]

hingga [15] terdapat fungsi ConstraintLayout digunakan sebagai layout utama di dalam CardView. ConstraintLayout dipilih karena fleksibilitasnya dalam mengatur posisi elemen UI dengan constraint antar komponen. Lebarnya diset match_parent agar memenuhi lebar CardView, sedangkan tingginya wrap_content, artinya akan menyesuaikan tinggi konten di dalamnya. Properti padding="25dp" memberikan ruang di dalam layout agar isi seperti teks dan gambar tidak menempel langsung ke tepi kartu, sehingga tampilan lebih rapi dan nyaman dilihat. Pada baris [17] hingga [27] terdapat fungsi ImageView dengan ID imgGunung yang digunakan untuk menampilkan gambar gunung dalam setiap item list. Ukuran gambar disetel sebesar 120dp lebar dan 160dp tinggi, dengan scaleType="centerCrop" agar gambar memenuhi ruang tanpa mengubah rasio secara tidak proporsional. Gambar ini ditempatkan di sisi kiri item dengan batas kanan (end) yang terhubung ke komponen linearLayoutText, dan diberi margin end sebesar 12dp agar tidak menempel langsung. Posisi atas dan kiri dikaitkan ke parent untuk memastikan gambar berada di bagian atas dan kiri dari kartu. Pada baris [29] hingga [38] terdapat fungsi LinearLayout dengan ID linearLayoutText ini digunakan untuk menampung elemen teks seperti nama gunung, lokasi, dan deskripsi secara vertikal di sebelah kanan gambar. Lebarnya disetel 0dp agar mengikuti aturan ConstraintLayout, dan tingginya wrap_content menyesuaikan isi. Komponen ini dikaitkan dengan sisi kanan (end) dari imgGunung, sisi atas parent, dan sisi kanan parent, sehingga menempati sisa ruang horizontal di samping gambar. Margin atas sebesar 8dp memberikan jarak dari atas, dan meskipun terdapat atribut layout_weight, atribut ini tidak berlaku di dalam ConstraintLayout sehingga bisa diabaikan atau dihapus. Pada baris [40] hingga [47] terdapat fungsi TextView dengan ID tvGunungName berfungsi untuk menampilkan nama gunung pada setiap item dalam daftar. Komponen ini memiliki lebar dan tinggi yang disesuaikan secara otomatis dengan isi teksnya, karena menggunakan atribut wrap_content. Ukuran teks diatur sebesar 24sp, cukup besar untuk menarik perhatian pengguna sebagai judul utama pada tampilan kartu. Gaya teks dibuat tebal (bold) agar terlihat lebih mencolok dan menonjol, serta menggunakan warna hitam (@android:color/black) agar kontrasnya jelas dengan latar belakang. Untuk keperluan pratinjau di Android Studio, diberikan teks contoh "Gunung Semeru" melalui atribut

tools:text, namun nilai ini tidak akan muncul saat aplikasi dijalankan. Desain ini membantu pengguna mengenali nama gunung secara cepat dan jelas pada tampilan daftar. Pada baris [49] hingga [56] terdapat fungsi TextView dengan ID tvGunungLokasi digunakan untuk menampilkan lokasi dari gunung yang sedang ditampilkan dalam item daftar. Elemen ini memiliki ukuran teks sebesar 14sp, cukup kecil namun masih terbaca dengan jelas, memberikan informasi sekunder setelah nama gunung. Warna teks diatur menjadi abu-abu gelap dengan kode warna #666666, yang menandakan bahwa informasi ini bukan informasi utama, namun tetap penting. Gaya teks ditampilkan dalam bentuk tebal (bold) untuk menambah penekanan dan keterbacaan. Seperti biasa, atribut tools:text diisi dengan teks contoh "Lokasi: Jawa Timur" untuk memberikan pratinjau saat mendesain di Android Studio, namun tidak memengaruhi teks saat runtime. TextView ini membantu pengguna mengetahui dengan cepat lokasi geografis gunung dalam daftar. Pada baris [58] hingga [69] terdapat fungsi TextView dengan ID tvGunungDeskripsi ini berfungsi untuk menampilkan deskripsi singkat mengenai gunung yang ditampilkan dalam setiap item daftar RecyclerView. Ukuran teksnya disesuaikan sebesar 14sp agar tetap mudah dibaca, sementara warna hitam (@android:color/black) digunakan untuk memastikan keterbacaan di latar belakang terang. Untuk menjaga tata letak tetap rapi dan tidak memakan terlalu banyak ruang, deskripsi ini dibatasi maksimal dua baris menggunakan atribut android:maxLines="2". Jika teks melebihi dua baris, maka akan dipotong dan ditandai dengan elipsis (...) di akhir melalui android:ellipsize="end". Penggunaan tools:text menampilkan contoh isi deskripsi saat proses desain, tanpa mempengaruhi tampilan aplikasi saat dijalankan. Pada baris [71] hingga [82] terdapat fungsi LinearLayout ini digunakan untuk menampung dua tombol aksi (misalnya tombol "Detail" dan "Link") yang ditampilkan secara **horizontal** di bagian bawah informasi gunung pada setiap item RecyclerView. Layout ini memiliki lebar penuh (match_parent) dan tinggi menyesuaikan kontennya (wrap_content), dengan orientasi horizontal agar anak-anaknya (biasanya dua tombol) tersusun sejajar ke kanan. Atribut android:gravity="end" mengarahkan konten ke sisi kanan dari LinearLayout. Penambahan android:layout_marginTop="8dp" memberi jarak vertikal dari elemen di atasnya

(teks), dan android:layout_marginStart="130dp" memberikan ruang kosong di sisi kiri agar posisi tombol tampak rata dengan bagian informasi teks di kanan gambar. Atribut weightSum="2" menandakan total bobot layout ini dibagi dua, sehingga masing-masing tombol di dalamnya dapat diberi layout_weight="1" untuk proporsi lebar yang seimbang. Constraint seperti app:layout_constraintStart_toStartOf="parent" dan app:layout_constraintTop_toBottomOf="@+id/linearLayoutText" menunjukkan bahwa LinearLayout ini masih berada dalam konteks ConstraintLayout, dan ditambatkan secara tepat di bawah elemen teks. Terakhir, pada baris [84] hingga [98] terdapat Dua tombol ini masing-masing diberi layout_width="0dp" dan layout_weight="1" agar keduanya membagi ruang secara merata di dalam LinearLayout yang memiliki weightSum="2". Tombol pertama, dengan id="@+id btnLink", memiliki teks "Link" dan layout_marginEnd="8dp" untuk memberi jarak horizontal antara dua tombol. Tombol kedua, btnDetail, memiliki teks "Detail". Keduanya memungkinkan pengguna untuk berinteraksi lebih lanjut: tombol "Detail" biasanya akan menampilkan informasi lengkap tentang gunung tersebut di fragment atau halaman baru, sementara tombol "Link" bisa diarahkan untuk membuka referensi eksternal, seperti artikel, website resmi, atau Google Maps.

9. list_fragment.xml

Kode pada kelas ini menampilkan layout dari kelas listfragment.kt. pada baris [1] hingga [8] terdapat fungsi onstraintLayout agar elemen UI bisa diatur secara fleksibel dengan constraint antar komponen. Atribut tools:context=".ListFragment" menunjukkan bahwa layout ini digunakan oleh kelas ListFragment. Dengan layout_width dan layout_height di-set ke match_parent, maka layout ini akan menempati seluruh ruang layar yang tersedia. Biasanya, di dalam layout ini akan ditambahkan sebuah RecyclerView untuk menampilkan daftar item seperti gunung, beserta komponen-komponen lain yang mungkin dibutuhkan seperti toolbar, filter, atau search bar. Pada baris [10] hingga [23] terdapat fungsi RecyclerView yang digunakan untuk menampilkan daftar gunung dalam aplikasi. Dengan ID rvGunung, komponen ini diatur agar menyesuaikan ukuran penuh layar dengan menggunakan

constraint ke seluruh sisi parent layout. Atribut clipToPadding diset ke false agar konten bisa tetap terlihat saat di-scroll melewati padding. Latar belakang diberi warna biru muda (#94C2EB) dan diberi padding sebesar 16dp untuk memberikan jarak antara tepi layar dengan item di dalamnya. Atribut tools:listItem="@layout/item_gunung" hanya digunakan sebagai referensi desain di Android Studio, menunjukkan bahwa setiap item yang ditampilkan berasal dari layout item_gunung.xml.

app\src\main\res\values

10. colors.xml

Kode kelas ini berguna untuk mendefinisikan dua warna khusus untuk digunakan dalam aplikasi Android kamu. Warna blue_500 dengan nilai heksadesimal #2196F3 merupakan warna biru terang yang sering digunakan sebagai warna utama (primary color) dalam tema Material Design. Sedangkan blue_700 dengan nilai #1976D2 adalah versi yang lebih gelap dari warna biru tersebut, biasanya dipakai sebagai primaryDark atau untuk elemen-elemen yang membutuhkan aksen warna lebih kuat. Kedua warna ini dapat dipanggil di layout XML atau dalam kode Kotlin menggunakan resource ID seperti @color/blue_500 atau R.color.blue_700.

11. string.xml

Pada baris [1] hingga baris [9] terdapat fungsi Resource yang mendefinisikan nama aplikasi sebagai "MODUL 3" melalui elemen <string name="app_name">. Selain itu, juga terdapat sebuah array string bernama gunung_name yang memuat daftar lima gunung terkenal di Indonesia: Gunung Bromo, Gunung Kerinci, Gunung Merapi, Gunung Merbabu, dan Gunung Rinjani. Array ini biasanya digunakan untuk menampilkan pilihan dalam komponen UI seperti Spinner, ListView, atau sebagai sumber data untuk list dinamis lainnya seperti RecyclerView. Pada baris [11] hingga [17] terdapat fungsi Array gunung_link berfungsi sebagai kumpulan tautan atau URL yang mengarah ke situs resmi masing-masing gunung yang ditampilkan dalam aplikasi. Setiap elemen dalam array ini berisi alamat website dari taman nasional atau pengelola resmi gunung seperti Bromo, Kerinci, Merapi, Merbabu, dan Rinjani. Tautan-tautan ini dimanfaatkan oleh aplikasi untuk memberikan akses langsung

kepada pengguna terhadap informasi yang lebih lengkap dan terpercaya, seperti jalur pendakian, peraturan taman nasional, informasi perizinan, serta kegiatan konservasi. Biasanya, array ini akan dihubungkan dengan tombol di tampilan daftar atau detail gunung, sehingga ketika pengguna menekan tombol "Link", aplikasi akan membuka browser dan menampilkan halaman web sesuai URL yang bersangkutan. Pada baris [19] hingga [25] terdapat fungsi array gunung_lokasi yang digunakan untuk menyimpan informasi mengenai lokasi geografis dari setiap gunung yang ditampilkan dalam aplikasi. Setiap item dalam array ini berkorespondensi dengan item yang ada di array gunung_name, sehingga posisi masing-masing lokasi akan cocok dengan nama gunung yang bersangkutan. Misalnya, posisi pertama berisi "Jawa Timur" yang merupakan lokasi dari Gunung Bromo, sementara posisi kedua berisi "Jambi, di Taman Nasional Kerinci Seblat" untuk Gunung Kerinci, dan seterusnya. Informasi ini ditampilkan di antarmuka pengguna untuk memberi gambaran singkat kepada pengguna mengenai letak tiap gunung. Dengan menyusun data lokasi dalam string-array, developer dapat dengan mudah mengelola, menampilkan, dan memperbarui data lokasi secara efisien melalui adapter RecyclerView atau fragment detail dalam aplikasi Android. Pada baris [27] hingga [125] yang diberikan mendefinisikan sebuah array string dengan nama gunung_deskripsi yang berisi beberapa item deskripsi tentang gunung. Array ini digunakan untuk menyimpan kumpulan teks atau informasi terkait gunung yang dapat diakses dalam aplikasi Android. Setiap elemen <item> di dalam array tersebut berisi deskripsi yang bisa digunakan untuk menampilkan informasi tentang gunung tertentu, seperti dalam tampilan daftar atau RecyclerView. Array ini memungkinkan aplikasi untuk mengorganisir dan mengelola beberapa teks deskripsi dengan lebih terstruktur, sehingga memudahkan dalam mengakses dan menampilkan informasi terkait gunung secara dinamis. Pada baris [130] hingga [136] terdapat fungsi yang mendefinisikan sebuah array dengan nama gunung_image yang berisi beberapa item gambar yang terletak di direktori drawable aplikasi Android. Setiap elemen <item> di dalam array ini merujuk pada gambar yang berbeda, seperti @drawable/gunung_bromo, @drawable/gunung_kerinci, dan seterusnya. Gambar-gambar tersebut akan digunakan untuk mewakili berbagai gunung yang ada dalam aplikasi.

12. theme.xml

Kode kelas ini digunakan untuk mendefinisikan tema untuk aplikasi Android dengan menggunakan styles.xml. Tema ini dimulai dengan mendeklarasikan Base.Theme.Modul3, yang merupakan tema dasar yang diwarisi dari Theme.Material3.DayNight.NoActionBar, memberikan aplikasi dukungan mode terang dan gelap (DayNight) tanpa action bar. Di dalam tema dasar ini, beberapa atribut warna ditentukan, seperti colorPrimary yang menggunakan warna biru (@color/blue_500), colorOnPrimary yang menggunakan warna putih (@android:color/white) untuk teks atau ikon di atas elemen utama, serta colorPrimaryContainer yang menggunakan warna biru lebih gelap (@color/blue_700) untuk latar belakang elemen utama. Atribut colorOnPrimaryContainer juga diatur ke warna putih untuk teks di atas latar belakang tersebut. Selanjutnya, tema Theme.Modul3 mewarisi pengaturan dari Base.Theme.Modul3 dan dapat dimodifikasi lebih lanjut sesuai kebutuhan aplikasi. Dengan struktur ini, tema memastikan aplikasi memiliki konsistensi warna yang estetis dan mengikuti desain Material dengan mudah, sementara juga memberikan fleksibilitas dalam pengaturan tampilan aplikasi.

13. AndroidManifest.kt

Pada baris [3] hingga [6] terdapat fungsi import, diantaranya LayoutInflater digunakan untuk mengubah layout XML menjadi objek View yang dapat ditampilkan di UI, sementara ViewGroup digunakan untuk mendefinisikan grup tampilan tempat item RecyclerView akan diletakkan. RecyclerView adalah komponen UI yang memungkinkan tampilan daftar item yang efisien dan dapat di-scroll. Kode ini juga mengimpor kelas ItemGunungBinding, yang dihasilkan otomatis oleh Android Studio ketika menggunakan ViewBinding. Dengan ViewBinding, kita bisa mengakses elemen-elemen UI dalam layout secara langsung tanpa perlu menggunakan findViewById, yang lebih aman dan mengurangi kemungkinan kesalahan. Dalam implementasi adapter RecyclerView, ItemGunungBinding digunakan untuk mengikat data ke tampilan item, memungkinkan akses langsung ke elemen UI yang ada dalam layout item_gunung.xml, seperti teks atau gambar, dengan

cara yang lebih sederhana dan efisien. Pada baris [8] hingga [13] terdapat fungsi yang digunakan untuk mendefinisikan kelas GunungAdapter, yang merupakan adapter untuk RecyclerView di aplikasi Android. Kelas ini mengelola dan menampilkan data dalam bentuk daftar yang terdiri dari objek Gunung. Di dalam konstruktor, terdapat tiga parameter: listGunung, yang merupakan ArrayList<Gunung> yang berisi data gunung yang akan ditampilkan, onLinkClick, yang adalah fungsi untuk menangani aksi klik pada link yang menerima parameter bertipe String, dan onDetailClick, yang adalah fungsi untuk menangani klik pada elemen yang menampilkan detail gunung, dengan parameter yang mencakup ID dan tiga string lainnya (mungkin nama, lokasi, dan deskripsi gunung). Kelas ini mewarisi RecyclerView.Adapter dan menggunakan ListViewHolder sebagai ViewHolder untuk mengikat tampilan item. Pada baris [15] hingga [30] fungsi ListViewHolder yang merupakan bagian dari adapter RecyclerView dan bertugas untuk mengikat data gunung ke tampilan item. Kelas ini menerima objek ItemGunungBinding, yang memungkinkan akses ke elemen-elemen UI dalam layout item secara langsung melalui ViewBinding. Di dalam fungsi bind(), data dari objek Gunung diikat ke elemen UI seperti nama gunung, lokasi, deskripsi, dan gambar. binding.tvGunungName.text diatur dengan nama gunung, binding.tvGunungLokasi.text diatur dengan lokasi, binding.tvGunungDeskripsi.text dengan deskripsi, dan binding.imgGunung.setImageResource digunakan untuk menampilkan gambar gunung berdasarkan resource ID. Selain itu, dua tombol diatur untuk menangani klik. Tombol binding.btnExit memiliki listener yang memanggil fungsi onLinkClick dengan parameter link gunung ketika diklik, sementara tombol binding.btnExit memiliki listener yang memanggil fungsi onDetailClick dengan parameter berupa informasi gunung seperti ID gambar, nama, lokasi, dan deskripsi ketika diklik. Pada baris [35] hingga [48] terdapat fungsi RecyclerView.Adapter yang bertugas untuk mengelola data dan menampilkan item di dalam RecyclerView. Fungsi onCreateViewHolder bertanggung jawab untuk membuat tampilan item dengan meng-inflate layout menggunakan ItemGunungBinding.inflate(), yang memungkinkan akses mudah ke elemen UI menggunakan ViewBinding. Fungsi ini kemudian mengembalikan objek ListViewHolder, yang akan mengikat tampilan ke data yang sesuai. Fungsi getItemCount mengembalikan jumlah total item yang ada

dalam daftar listGunung, yang memberi tahu RecyclerView berapa banyak item yang perlu ditampilkan. Sementara itu, fungsi onBindViewHolder dipanggil untuk mengikat data ke tampilan. Di dalam fungsi ini, data dari objek Gunung pada posisi tertentu diambil dan diteruskan ke metode bind() dalam ViewHolder, yang akan memperbarui elemen-elemen UI dengan informasi yang relevan. Selain itu, fungsi onLinkClick dan onDetailClick diteruskan untuk menangani interaksi pengguna seperti klik pada tombol.

14. BuilGradle.kts

Kode kelas ini secara keseluruhan berguna untuk engatur konfigurasi dan dependensi aplikasi. Pada baris [1] hingga [4] terdapat fungsi plugins mendeklarasikan plugin yang digunakan, yaitu plugin Android aplikasi, Kotlin Android, dan kotlin-parcelize untuk mempermudah pengiriman objek antar-komponen dengan Parcelable. Pada baris [7] hingga [19] terdapat fungsi android, ditentukan berbagai konfigurasi seperti namespace, versi compileSdk, serta pengaturan defaultConfig yang mencakup applicationId, minimum dan target SDK, versi aplikasi, serta runner untuk pengujian instrumentasi. Pada baris [22] hingga [28] terdapat fungsi buildTypes berisi konfigurasi untuk mode rilis, termasuk pengaturan ProGuard untuk optimisasi dan obfuscation kode. Pada baris [33] hingga [39] terdapat fungsi compileOptions dan kotlinOptions menentukan bahwa proyek ini menggunakan Java 17 dan Kotlin dengan target JVM 17, menyesuaikan dengan fitur-fitur modern. Pada baris [42] dan [43] terdapat fungsi buildFeatures yang diatur untuk mengaktifkan viewBinding, yang memudahkan akses elemen layout dalam kode Kotlin. Terakhir, pada baris [47] hingga [64] terdapat fungsi dependencies yang mencantumkan pustaka-pustaka penting yang digunakan dalam proyek, seperti core-ktx, appcompat, material, constraintlayout, recyclerview, cardview, dan lifecycle components. Selain itu, terdapat dependensi untuk unit testing (junit) dan pengujian instrumentasi (espresso dan androidx.test).

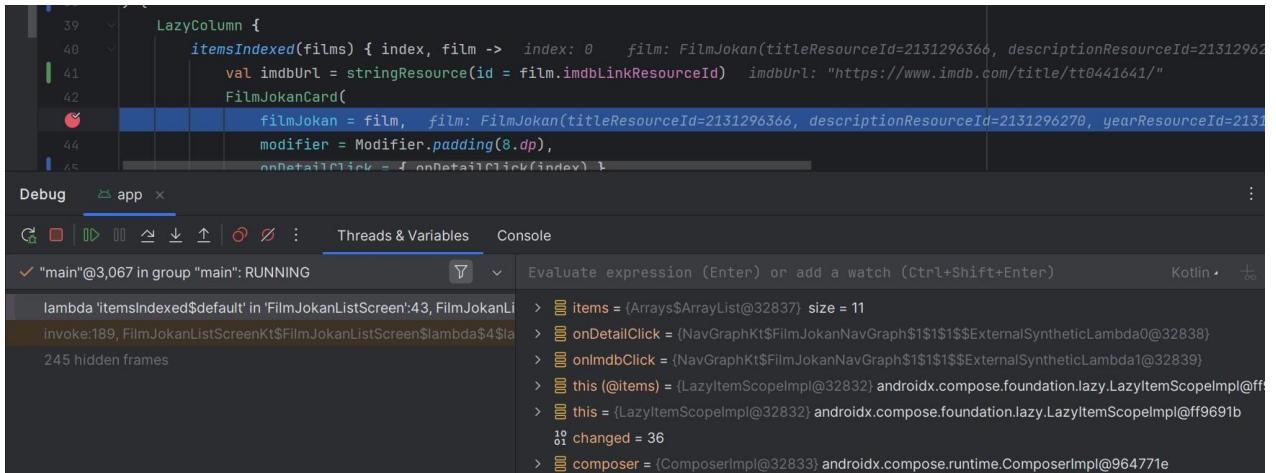
MODUL 4 : ViewModel and Debugging

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

- a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
- b. Gunakan ViewModelFactory dalam pembuatan ViewModel.
- c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
- d. Gunakan logging untuk event berikut:
 - a. Log saat data item masuk ke dalam list
 - b. Log saat tombol Detail dan tombol Explicit Intent ditekan
 - c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
- e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Intro, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out. Source Code

Aplikasi harus dapat mempertahankan fitur – fitur yang sudah dibuat pada modul sebelumnya. Berikut adalah contoh debugging dalam Android Studio.



Gambar 1. Contoh Penggunaan Debugger

A. Source Code

app\src\main\java\com\example\modul4

1. DetailFragment.kt

Tabel 20. Source Code Soal 1 Modul 4

```
1 package com.example.modul4
2
3 import android.os.Bundle
4 import androidx.fragment.app.Fragment
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.view.ViewGroup
8 import com.example.modul3.databinding.DetailFragmentBinding
9
10
11 class DetailFragment : Fragment() {
12
13     private var _binding: DetailFragmentBinding? = null
14     private val binding get() = _binding!!
15
16     override fun onCreateView(
17         inflater: LayoutInflater, container: ViewGroup?,
18         savedInstanceState: Bundle?
19     ): View {
20         _binding = DetailFragmentBinding.inflate(inflater,
21         container, false)
22
23         val image = arguments?.getInt("EXTRA_PHOTO")
24         val name = arguments?.getString("EXTRA_NAME")
25         val lokasi = arguments?.getString("EXTRA_LOKASI")
26         val deskripsi =
27             arguments?.getString("EXTRA_DESKRIPSI")
28
29         binding.tvName.text = name
30         binding.tvLokasi.text = lokasi
31         binding.tvDeskripsi.text = deskripsi
32         image?.let {
33             binding.imgPoster.setImageResource(it)
34         }
35     }
36
37 }
```

```
32         return binding.root
33     }
34
35     override fun onDestroyView() {
36         super.onDestroyView()
37         _binding = null
38     }
39 }
```

2. Gunung.kt

Tabel 21. Source Code Gunung.kt

```
1 package com.example.modul4
2
3 import android.os.Parcelable
4 import kotlinx.parcelize.Parcelize
5
6 @Parcelize
7 data class Gunung(
8     val image: Int,
9     val name: String,
10    val lokasi: String,
11    val deskripsi: String,
12    val link: String
13
14
15 ): Parcelable
```

3. GunungAdapter.kt

Tabel 22. Source Code Gunung.kt Soal 1 Modul 4

```
1 package com.example.modul4
2
3 import android.view.LayoutInflater
```

```
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.DiffUtil
6 import androidx.recyclerview.widget.ListAdapter
7 import androidx.recyclerview.widget.RecyclerView
8 import com.example.modul4.databinding.ItemGunungBinding
9
10 class GunungAdapter(
11     private val onLinkClick: (Gunung) -> Unit,
12     private val onDetailClick: (Gunung) -> Unit
13 ) : ListAdapter<Gunung,
14 GunungAdapter.ListViewHolder>(DIFF_CALLBACK) {
15
16     inner class ListViewHolder(private val binding:
17 ItemGunungBinding) :
18         RecyclerView.ViewHolder(binding.root) {
19
20         fun bind(gunung: Gunung) {
21             binding.tvGunungName.text = gunung.name
22             binding.tvGunungLokasi.text = gunung.lokasi
23             binding.tvGunungDeskripsi.text = gunung.deskripsi
24             binding.imgGunung.setImageResource(gunung.image)
25
26             binding.btnLink.setOnClickListener {
27                 onLinkClick(gunung)
28             }
29             binding.btnDetail.setOnClickListener {
30                 onDetailClick(gunung)
31             }
32         }
33     }
34
35     override fun onCreateViewHolder(parent: ViewGroup,
36 viewType: Int): ListViewHolder {
37         val binding =
```

```

38     ItemGunungBinding.inflate(LayoutInflater.from(parent.context),
39     parent, false)
40
41     return ListViewHolder(binding)
42
43     override fun onBindViewHolder(holder: ListViewHolder,
44     position: Int) {
45
46         holder.bind(getItem(position))
47
48     companion object {
49
50         private val DIFF_CALLBACK = object :
51 DiffUtil.ItemCallback<Gunung>() {
52
53             override fun areItemsTheSame(oldItem: Gunung,
54             newItem: Gunung) =
55
56                 oldItem.name == newItem.name
57
58             override fun areContentsTheSame(oldItem: Gunung,
59             newItem: Gunung) =
60
61                 oldItem == newItem
62
63         }
64     }
65 }

```

4. GunungViewModel.kt

Tabel 23. Source Code GunungViewModel.kt Soal 1 Modul 4

```

1 package com.example.modul4.viewmodel
2
3 import android.util.Log
4 import androidx.lifecycle.ViewModel
5 import androidx.lifecycle.viewModelScope
6 import com.example.modul4.Gunung
7 import kotlinx.coroutines.flow.MutableStateFlow

```

```
8 import kotlinx.coroutines.flow.StateFlow
9 import kotlinx.coroutines.launch
10
11 class GunungViewModel(private val initialList: List<Gunung>) :
12 ViewModel() {
13
14     private val _gunungList =
15 MutableStateFlow<List<Gunung>>(emptyList())
16     val gunungList: StateFlow<List<Gunung>> get() = _gunungList
17
18     private val _selectedGunung =
19 MutableStateFlow<Gunung?>(null)
20     val selectedGunung: StateFlow<Gunung?> get() =
21 _selectedGunung
22
23     init {
24         viewModelScope.launch {
25             _gunungList.value = initialList
26             Log.d("GunungViewModel", "Data gunung dimasukkan:
27 ${initialList.size} item")
28         }
29     }
30
31     fun onDetailClicked(gunung: Gunung) {
32         _selectedGunung.value = gunung
33         Log.d("GunungViewModel", "Detail diklik:
34 ${gunung.name}")
35     }
36
37     fun onLinkClicked(gunung: Gunung) {
38         Log.d("GunungViewModel", "Explicit intent diklik untuk:
39 ${gunung.name}")
40     }
41 }
```

5. GunungViewModelFactory.kt

Tabel 24. Source Code GunungViewModelFactory.kt Soal 1 Modul 4

```
1 package com.example.modul4.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import com.example.modul4.Gunung
6
7 class GunungViewModelFactory(private val gunungList:
8 List<Gunung>) : ViewModelProvider.Factory {
9     override fun <T : ViewModel> create(modelClass: Class<T>):
10 T {
11         if
12             (modelClass.isAssignableFrom(GunungViewModel::class.java)) {
13                 @Suppress("UNCHECKED_CAST")
14                 return GunungViewModel(gunungList) as T
15             }
16             throw IllegalArgumentException("Unknown ViewModel
17 class")
18     }
19 }
```

6. ListFragment.kt

Tabel 25. Source Code ListFragment.kt Soal 1 Modul 4

```
1 package com.example.modul4
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.os.Bundle
6 import android.util.Log
7 import androidx.fragment.app.Fragment
```

```
8 import android.view.LayoutInflater
9 import android.view.View
10 import android.view.ViewGroup
11 import androidx.lifecycle.ViewModelProvider
12 import androidx.lifecycle.lifecycleScope
13 import androidx.recyclerview.widget.LinearLayoutManager
14 import com.example.modul4.databinding.ListFragmentBinding
15 import com.example.modul4.viewmodel.GunungViewModelFactory
16 import com.example.modul4.viewmodel.GunungViewModel
17
18
19 class ListFragment : Fragment() {
20
21     private var _binding: ListFragmentBinding? = null
22     private val binding get() = _binding!!
23
24     private lateinit var viewModel: GunungViewModel
25
26     override fun onCreateView(
27         inflater: LayoutInflater, container: ViewGroup?,
28         savedInstanceState: Bundle?
29     ): View {
30         _binding = ListFragmentBinding.inflate(inflater,
31         container, false)
32
33         val listGunung = getListGunung()
34         val factory = GunungViewModelFactory(listGunung)
35         viewModel = ViewModelProvider(this,
36         factory)[GunungViewModel::class.java]
37
38         setupRecyclerView()
39         observeViewModel()
40
41         return binding.root
```

```
42    }
43
44    private fun setupRecyclerView() {
45        val adapter = GunungAdapter(
46            onLinkClick = { gunung ->
47                viewModel.onLinkClicked(gunung)
48                val intent = Intent(Intent.ACTION_VIEW,
49 Uri.parse(gunung.link))
50                startActivity(intent)
51            },
52            onDetailClick = { gunung ->
53                viewModel.onDetailClicked(gunung)
54                val detailFragment = DetailFragment().apply {
55                    arguments = Bundle().apply {
56                        putInt("EXTRA_PHOTO", gunung.image)
57                        putString("EXTRA_NAME", gunung.name)
58                        putString("EXTRA_LOKASI",
59 gunung.lokasi)
60                        putString("EXTRA_DESKRIPSI",
61 gunung.deskripsi)
62                    }
63                }
64                parentFragmentManager.beginTransaction()
65                    .replace(R.id.frame_container,
66 detailFragment)
67                    .addToBackStack(null)
68                    .commit()
69            }
70        )
71        binding.rvGunung.layoutManager =
72 LinearLayoutManager(requireContext())
73        binding.rvGunung.adapter = adapter
74        binding.rvGunung.setHasFixedSize(true)
75    }
```

```
76         lifecycleScope.launchWhenStarted {
77             viewModel.gunungList.collect { list ->
78                 adapter.submitList(list)
79             }
80         }
81     }
82
83     private fun observeViewModel() {
84         lifecycleScope.launchWhenStarted {
85             viewModel.selectedGunung.collect { selected ->
86                 selected?.let {
87                     Log.d("ListFragment", "Navigasi ke detail:
88 ${it.name}")
89                 }
90             }
91         }
92     }
93
94     private fun getListGunung(): List<Gunung> {
95         val dataImage =
96             resources.obtainTypedArray(R.array.gunung_image)
97         val dataName =
98             resources.getStringArray(R.array.gunung_name)
99         val dataLokasi =
100            resources.getStringArray(R.array.gunung_lokasi)
101         val dataDesc =
102            resources.getStringArray(R.array.gunung_deskripsi)
103         val dataLink =
104            resources.getStringArray(R.array.gunung_link)
105
106         return List(dataName.size) { i ->
107             Gunung(
108                 dataImage.getResourceId(i, -1),
109                 dataName[i],
```

```

110         dataLokasi[i],
111         dataDesc[i],
112         dataLink[i]
113     )
114     }.also { dataImage.recycle() }
115 }
116
117 override fun onDestroyView() {
118     super.onDestroyView()
119     _binding = null
120 }
121 }
```

7. MainActivity.kt

Tabel 26. Source Code MainActivity.kt Soal 1 Modul 4

```

1 package com.example.modul4
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5
6
7 class MainActivity : AppCompatActivity() {
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11
12        val fragmentManager = supportFragmentManager
13        val listFragment = ListFragment()
14        val fragment =
15        fragmentManager.findFragmentByTag(ListFragment::class.java.simpleName)
16
17        if (fragment !is ListFragment) {
18            fragmentManager
```

```

19         .beginTransaction()
20             .add(R.id.frame_container, listFragment,
21 ListFragment::class.java.simpleName)
22             .commit()
23     }
24 }
25 }
```

app\src\main\res\layout

8. activity_main.xml

Tabel 27. Source Code activity_main.xml Soal 1 Modul 4

```

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

9. detail_fragment.xml

Tabel 28. Source Code detail_fragment.xml Soal 1 Modul 4

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:tools="http://schemas.android.com/tools"
5     xmlns:app="http://schemas.android.com/apk/res-auto"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="#B6D6F1"
```

```
9     android:padding="16dp"
10    tools:context=".DetailFragment">
11
12    <androidx.constraintlayout.widget.ConstraintLayout
13        android:layout_width="match_parent"
14        android:layout_height="wrap_content">
15
16        <ImageView
17            android:id="@+id/imgPoster"
18            android:layout_width="0dp"
19            android:layout_height="300dp"
20            android:contentDescription="Foto Gunung"
21            app:layout_constraintTop_toTopOf="parent"
22            app:layout_constraintStart_toStartOf="parent"
23            app:layout_constraintEnd_toEndOf="parent" />
24
25        <TextView
26            android:id="@+id/tvName"
27            android:layout_width="0dp"
28            android:layout_height="wrap_content"
29            android:gravity="center"
30            android:textSize="35sp"
31            android:textStyle="bold"
32            android:textColor="@android:color/black"
33            tools:text="Gunung Semeru"
34            android:layout_marginTop="12dp"
35            app:layout_constraintTop_toBottomOf="@+id/imgPoster"
36            app:layout_constraintStart_toStartOf="parent"
37            app:layout_constraintEnd_toEndOf="parent" />
38
39        <TextView
40            android:id="@+id/tvLokasi"
41            android:layout_width="0dp"
42            android:layout_height="wrap_content"
```

```
43         android:gravity="center"
44         android:textSize="22sp"
45         tools:text="Lokasi: Jawa Timur"
46         android:layout_marginTop="8dp"
47         app:layout_constraintTop_toBottomOf="@+id/tvName"
48         app:layout_constraintStart_toStartOf="parent"
49         app:layout_constraintEnd_toEndOf="parent" />
50
51     <TextView
52         android:id="@+id/tvDeskripsi"
53         android:layout_width="0dp"
54         android:layout_height="wrap_content"
55         android:textSize="18sp"
56         android:justificationMode="inter_word"
57         tools:text="Gunung tertinggi di Pulau Jawa yang
58 terkenal dengan jalur pendakian yang menantang dan keindahan
59 pemandangan alam."
60         android:layout_marginTop="8dp"
61         app:layout_constraintTop_toBottomOf="@+id/tvLokasi"
62         app:layout_constraintStart_toStartOf="parent"
63         app:layout_constraintEnd_toEndOf="parent" />
64
65     <Button
66         android:id="@+id/btnBack"
67         android:layout_width="wrap_content"
68         android:layout_height="wrap_content"
69         android:text="Kembali"
70         android:layout_marginTop="16dp"
71
72     app:layout_constraintTop_toBottomOf="@+id/tvDeskripsi"
73         app:layout_constraintStart_toStartOf="parent"
74         app:layout_constraintEnd_toEndOf="parent" />
75
76
```

```
77    </androidx.constraintlayout.widget.ConstraintLayout>
78
79</ScrollView>
```

10. item_gunung.xml

Tabel 29. Source Code item_gunung.xml Soal 1 Modul 4

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.cardview.widget.CardView
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="wrap_content"
8      android:layout_margin="8dp"
9      app:cardElevation="4dp"
10     app:cardCornerRadius="8dp">
11
12     <androidx.constraintlayout.widget.ConstraintLayout
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:padding="25dp">
16
17         <ImageView
18             android:id="@+id/imgGunung"
19             android:layout_width="120dp"
20             android:layout_height="160dp"
21             android:scaleType="centerCrop"
22             tools:src="@drawable/gunung_merbabu"
23             app:layout_constraintStart_toStartOf="parent"
24             app:layout_constraintTop_toTopOf="parent"
25
26             app:layout_constraintEnd_toStartOf="@+id/linearLayoutText"
27             android:layout_marginEnd="12dp"/>
28
```

```
29      <LinearLayout
30          android:id="@+id/linearLayoutText"
31          android:orientation="vertical"
32          android:layout_width="0dp"
33          android:layout_height="wrap_content"
34          app:layout_constraintStart_toEndOf="@+id/imgGunung"
35          app:layout_constraintTop_toTopOf="parent"
36          app:layout_constraintEnd_toEndOf="parent"
37          android:layout_marginTop="8dp"
38          android:layout_weight="1">
39
40      <TextView
41          android:id="@+id/tvGunungName"
42          android:layout_width="wrap_content"
43          android:layout_height="wrap_content"
44          android:textSize="24sp"
45          android:textStyle="bold"
46          android:textColor="@android:color/black"
47          tools:text="Gunung Semeru" />
48
49      <TextView
50          android:id="@+id/tvGunungLokasi"
51          android:layout_width="wrap_content"
52          android:layout_height="wrap_content"
53          android:textSize="14sp"
54          android:textColor="#666666"
55          android:textStyle="bold"
56          tools:text="Lokasi: Jawa Timur" />
57
58      <TextView
59          android:id="@+id/tvGunungDeskripsi"
60          android:layout_width="wrap_content"
61          android:layout_height="wrap_content"
62          android:textSize="14sp"
```

```
63         android:textColor="@android:color/black"
64         android:maxLines="2"
65         android:ellipsize="end"
66         tools:text="Gunung tertinggi di Jawa Timur
67 dengan pemandangan yang sangat indah dan memiliki trek yang
68 menantang untuk para pendaki pemula maupun profesional." />
69     </LinearLayout>
70
71     <LinearLayout
72         android:orientation="horizontal"
73         android:layout_width="match_parent"
74         android:layout_height="wrap_content"
75         android:gravity="end"
76         android:layout_marginTop="8dp"
77         android:layout_marginStart="130dp"
78         app:layout_constraintStart_toStartOf="parent"
79         app:layout_constraintEnd_toEndOf="parent"
80
81     app:layout_constraintTop_toBottomOf="@+id/linearLayoutText"
82         android:weightSum="2">
83
84         <Button
85             android:id="@+id/btnLink"
86             android:layout_width="0dp"
87             android:layout_height="wrap_content"
88             android:layout_weight="1"
89             android:text="Link"
90             android:layout_marginEnd="8dp" />
91
92         <Button
93             android:id="@+id/btnDetail"
94             android:layout_width="0dp"
95             android:layout_height="wrap_content"
96             android:layout_weight="1"
```

```

97         android:text="Detail" />
98     
```

- 99
- 100
- 101
- 102 </androidx.constraintlayout.widget.ConstraintLayout>
- 103 </androidx.cardview.widget.CardView>

11. list_fragment.xml

Tabel 30. Source Code list_fragment.xml Soal 1 Modul 4

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".ListFragment">
9
10    <androidx.recyclerview.widget.RecyclerView
11        android:id="@+id/rvGunung"
12        android:layout_width="0dp"
13        android:layout_height="0dp"
14        android:clipToPadding="false"
15        android:background="#94C2EB"
16        android:padding="16dp"
17        app:layout_constraintTop_toTopOf="parent"
18        app:layout_constraintBottom_toBottomOf="parent"
19        app:layout_constraintStart_toStartOf="parent"
20        app:layout_constraintEnd_toEndOf="parent"
21        tools:listitem="@layout/item_gunung" />
22    </androidx.constraintlayout.widget.ConstraintLayout>
23

```

app\src\main\res\values

12. colors.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="blue_500">#2196F3</color>
4     <color name="blue_700">#1976D2</color>
5 </resources>
6
```

Tabel 31. Source Code colors.xml

13. string.xml

Tabel 32. Source Code string.xml Soal 1 Modul 4

```
1 <resources>
2     <string name="app_name">MODUL 3</string>
3     <string-array name="gunung_name">
4         <item>Gunung Bromo</item>
5         <item>Gunung Kerinci</item>
6         <item>Gunung Merapi</item>
7         <item>Gunung Merbabu</item>
8         <item>Gunung Rinjani</item>
9     </string-array>
10
11    <string-array name="gunung_link">
12        <item>https://bromotenggersemeru.org/</item>
13        <item>https://tnkerinciseblat.com/</item>
14        <item>https://tngmerapi.id/</item>
15        <item>https://tngunungmerbabu.org/</item>
16        <item>https://www.rinjanationalpark.id/</item>
17    </string-array>
18
19    <string-array name="gunung_lokasi">
20        <item>Jawa Timur</item>
21        <item>Jambi, di Taman Nasional Kerinci Seblat</item>
```

```
22     <item>Perbatasan Yogyakarta dan Jawa Tengah</item>
23     <item>Jawa Tengah</item>
24     <item>Lombok, Nusa Tenggara Barat</item>
25   </string-array>
26
27   <string-array name="gunung_deskripsi">
28     <item>Gunung Bromo adalah gunung api aktif yang
29 terkenal secara internasional karena lanskapnya yang ikonik
30 dan mudah diakses dengan ketinggian 2.329 mdpl. Gunung ini
31 merupakan bagian dari kaldera Tengger yang sangat luas, dengan
32 lautan pasir (Segara Wedi) selebar sekitar 10 km yang
33 mengelilinginya. Kawah Bromo masih mengeluarkan asap putih dan
34 terkadang belerang, menjadikannya gunung yang terus dipantau
35 meskipun menjadi destinasi wisata utama. Bromo memiliki peran
36 penting dalam budaya masyarakat Suku Tengger, keturunan dari
37 kerajaan Majapahit, yang memegang teguh tradisi Hindu. Salah
38 satu upacara besar mereka adalah Yadnya Kasada, di mana mereka
39 melemparkan hasil panen, ternak, dan sesajen lainnya ke kawah
40 Bromo sebagai persembahan kepada para dewa. Upacara ini
41 menarik banyak wisatawan setiap tahun. Akses menuju Bromo
42 cukup mudah, terutama dari Cemoro Lawang (Probolinggo) dan
43 Wonokitri (Pasuruan). Wisatawan biasanya menantikan momen
44 matahari terbit dari Penanjakan, sebuah titik pandang di sisi
45 timur kaldera yang menawarkan panorama menakjubkan, dengan
46 Gunung Bromo, Batok, dan Semeru berjejer dalam satu garis
47 cakrawala.</item>
48     <item>Gunung Kerinci adalah puncak tertinggi di Pulau
49 Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan
50 total ketinggian 3.805 mdpl. Gunung ini berdiri megah di
51 tengah kawasan konservasi Taman Nasional Kerinci Seblat
52 (TNKS), yang juga merupakan situs Warisan Dunia UNESCO dalam
53 kategori Tropical Rainforest Heritage of Sumatra. Kawasan
54 sekitar Kerinci adalah salah satu titik keanekaragaman hayati
55 terkaya di dunia, menjadi rumah bagi harimau Sumatera, tapir,
```

56 beruang madu, dan banyak jenis burung endemik. Kerinci masih
57 aktif dan kerap menunjukkan aktivitas vulkanik berupa letusan
58 kecil, gempa vulkanik, dan hembusan asap kawah. Meskipun
59 begitu, gunung ini tetap menjadi daya tarik pendakian bagi
60 pencinta alam ekstrem. Jalur utama pendakian melalui desa
61 Kersik Tuo memiliki trek yang panjang dan menantang, melewati
62 hutan hujan tropis lebat, rawa, dan jalur batu curam. Dari
63 puncaknya, pendaki dapat melihat garis pantai Samudera Hindia
64 dan Pegunungan Bukit Barisan yang menyapu cakrawala. Selain
65 sebagai objek wisata, Kerinci juga penting untuk penelitian
66 geologi dan pelestarian ekosistem pegunungan tropis.</item>
67 <item>Gunung Merapi merupakan gunung berapi paling
68 aktif di Indonesia dan salah satu yang paling aktif di dunia
69 dengan total ketinggian 2.930 mdpl. Merapi secara rutin
70 mengalami letusan setiap 2-5 tahun sekali dan sangat
71 memengaruhi wilayah padat penduduk di sekitarnya. Letusan
72 besar terakhir yang menyebabkan korban jiwa terjadi pada tahun
73 2010, menewaskan puluhan orang dan memaksa ribuan lainnya
74 mengungsi. Merapi bukan sekadar gunung, tetapi juga simbol
75 budaya dan spiritualitas masyarakat Jawa. Gunung ini dipercaya
76 sebagai pusat dunia spiritual dalam kosmologi Keraton
77 Yogyakarta. Setiap tahun, masyarakat melakukan ritual Labuhan
78 Merapi sebagai bentuk persembahan dan penghormatan terhadap
79 kekuatan alam. Selain itu, kawasan lereng Merapi menjadi objek
80 wisata edukasi, seperti Museum Gunungapi Merapi dan tur lava
81 jeep yang memperlihatkan bekas jalur aliran awan panas (wedhus
82 gembel). Secara geologis, Merapi terus dimonitor secara
83 intensif oleh PVMBG dan BPPTKG dengan berbagai alat modern
84 seperti seismograf, kamera thermal, dan satelit. Pendaki
85 umumnya hanya diperbolehkan naik hingga Pasar Bubrah, karena
86 puncaknya sangat berisiko terkena guguran lava dan awan
87 panas..</item>
88 <item>Gunung Merbabu adalah gunung dengan ketinggian
89 3.145 meter di atas permukaan laut (mdpl) yang bertipe

90 stratovolcano yang sudah tidak aktif, berdampingan erat dengan
91 Gunung Merapi di sebelah selatannya. Nama "Merbabu" berasal
92 dari gabungan kata "Meru" (gunung) dan "Abu", yang secara
93 harfiah berarti "gunung abu". Gunung ini memiliki keunikan
94 berupa hamparan padang sabana luas yang sangat memikat,
95 terutama saat musim kemarau ketika rerumputan menguning
96 keemasan. Pendaki umumnya memilih jalur via Selo (Boyolali)
97 atau Wekas (Magelang) karena pemandangan spektakuler dan trek
98 yang relatif ramah. Gunung Merbabu sangat populer di kalangan
99 pendaki, terutama karena spot sunrise dari puncaknya yang
100 memperlihatkan lanskap gunung lain seperti Merapi, Sumbing,
101 Sindoro, Lawu, dan bahkan Slamet. Flora dan fauna di kawasan
102 ini cukup beragam, termasuk edelweiss, burung jalak, serta
103 lutung Jawa. Meskipun secara vulkanik tidak aktif, Merbabu
104 tetap diawasi karena posisinya yang dekat dengan Merapi yang
105 sangat aktif. Selain pendakian, wilayah di kaki Merbabu juga
106 dimanfaatkan untuk pertanian hortikultura oleh warga
107 lokal.</item>
108 <item>Gunung Rinjani merupakan gunung berapi tertinggi
109 kedua di Indonesia dan menjadi ikon Pulau Lombok dengan
110 ketinggian 3.726 mdpl. Gunung ini termasuk dalam Taman
111 Nasional Gunung Rinjani yang luasnya mencapai lebih dari
112 41.000 hektare. Di kawahnya terdapat Danau Segara Anak, yang
113 menjadi pusat spiritual dan simbol kehidupan bagi masyarakat
114 lokal. Di tengah danau tersebut, menjulang Gunung Barujari,
115 yang merupakan kawah aktif dari Rinjani dan menjadi sumber
116 letusan terakhir pada 2016. Pendakian Rinjani dikenal berat
117 dan membutuhkan stamina serta persiapan fisik yang matang.
118 Jalur-jalur populer antara lain via Sembalun (lebih terbuka
119 dan panas) serta Senaru (lebih rimbun dan teduh). Rinjani
120 menawarkan pemandangan yang menakjubkan: dari hutan tropis,
121 air terjun, danau, hingga puncak berbatu yang curam. Selain
122 nilai geologis, Rinjani juga memiliki arti spiritual dan
123 budaya tinggi, seperti ritual Pekelan atau Mulang Pakelem yang

```

124 dilakukan masyarakat Bali dan Sasak di Danau Segara Anak
125 sebagai bentuk penghormatan kepada alam.
126
127 </item>
128     </string-array>
129
130     <array name="gunung_image">
131         <item>@drawable/gunung_bromo</item>
132         <item>@drawable/gunung_kerinci</item>
133         <item>@drawable/gunung_merapi</item>
134         <item>@drawable/gunung_merbabu</item>
135         <item>@drawable/gunung_rinjani</item>
136     </array>
137 </resources>

```

14. theme.xml

Tabel 33. Source Code theme.xml Soal 1 Modul 4

```

1 <resources xmlns:tools="http://schemas.android.com/tools">
2     <!-- Base application theme. -->
3     <style name="Base.Theme.Modul3"
4 parent="Theme.Material3.DayNight.NoActionBar">
5         <!-- Tambahkan warna utama -->
6         <item name="colorPrimary">@color/blue_500</item>
7         <item name="colorOnPrimary">@android:color/white</item>
8         <item
9             name="colorPrimaryContainer">@color/blue_700</item>
10            <item
11                name="colorOnPrimaryContainer">@android:color/white</item>
12            </style>
13
14     <style name="Theme.Modul3" parent="Base.Theme.Modul3" />
15 </resources>

```

app\src\main

15. AndroidManifest.xml

Tabel 34. Source Code AndroidManifest.xml Soal 1 Modul 4

```
1 package com.example.modul3
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.example.modul3.databinding.ItemGunungBinding
7
8 class GunungAdapter(
9     private val listGunung: ArrayList<Gunung>,
10    private val onLinkClick: (String) -> Unit,
11    private val onDetailClick: (Int, String, String, String) ->
12 Unit
13 ) : RecyclerView.Adapter<GunungAdapter.ListViewHolder>() {
14
15     class ListViewHolder(private val binding:
16 ItemGunungBinding) : RecyclerView.ViewHolder(binding.root) {
17         fun bind(gunung: Gunung, onLinkClick: (String) -> Unit,
18 onDetailClick: (Int, String, String, String) -> Unit) {
19             // Bind data to views using the binding object
20             binding.tvGunungName.text = gunung.name
21             binding.tvGunungLokasi.text = gunung.lokasi
22             binding.tvGunungDeskripsi.text = gunung.deskripsi
23             binding.imgGunung.setImageResource(gunung.image)
24
25             binding.btnLink.setOnClickListener {
26                 onLinkClick(gunung.link)
27
28             binding.btnDetail.setOnClickListener {
29                 onDetailClick(gunung.image, gunung.name,
30                 gunung.lokasi, gunung.deskripsi)
```

```

31        }
32    }
33 }
34
35     override fun onCreateViewHolder(parent: ViewGroup,
36 viewType: Int): ListViewHolder {
37         val binding =
38 ItemGunungBinding.inflate(LayoutInflater.from(parent.context),
39 parent, false)
40         return ListViewHolder(binding)
41     }
42
43     override fun getItemCount(): Int = listGunung.size
44
45     override fun onBindViewHolder(holder: ListViewHolder,
46 position: Int) {
47         val gunung = listGunung[position]
48         holder.bind(gunung, onLinkClick, onDetailClick)
49     }
50 }
```

16. build.gradle.kts

Tabel 35. Source Code BuildGradle.kts Soal 1 Modul 4

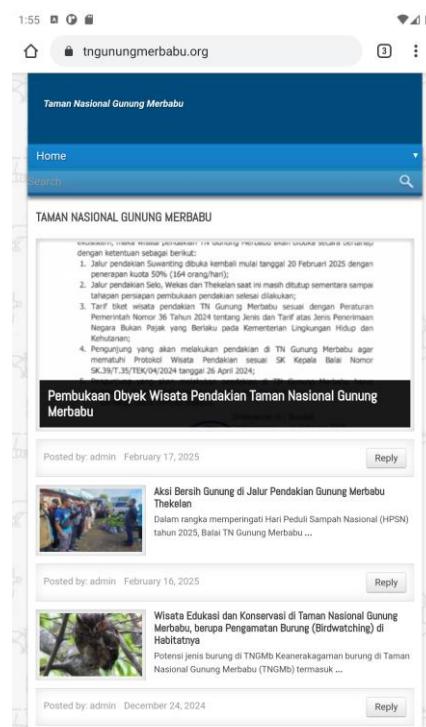
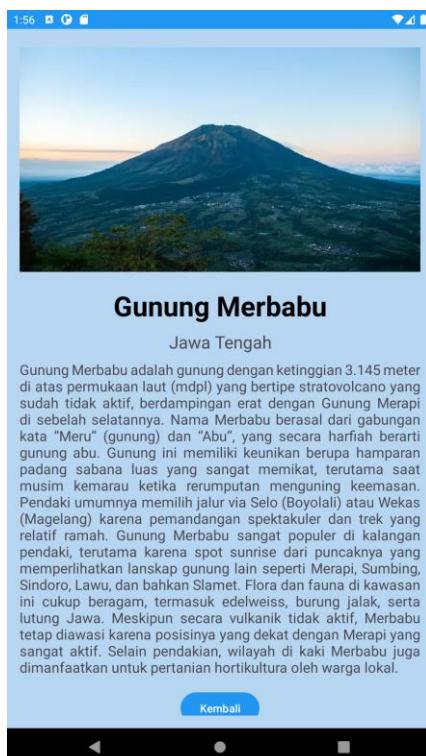
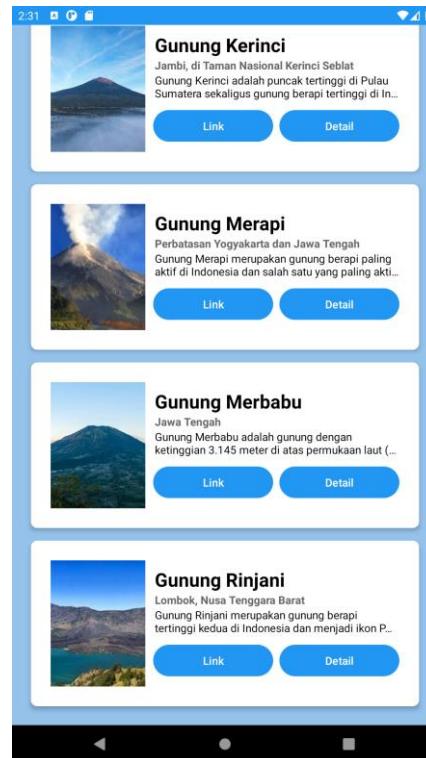
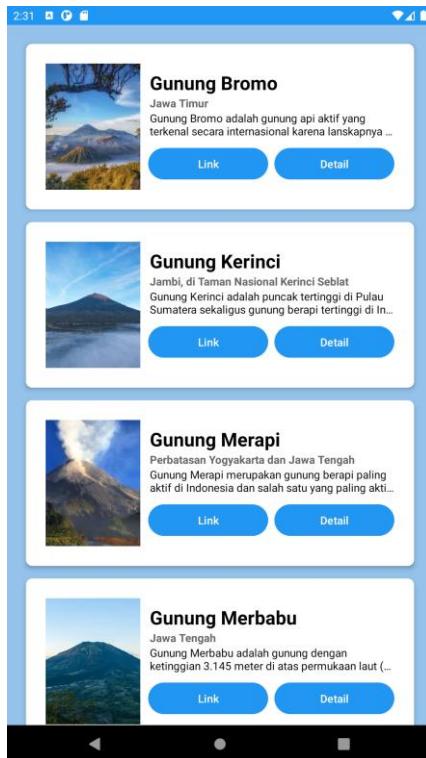
```

1 plugins {
2     id("com.android.application")
3     id("org.jetbrains.kotlin.android")
4     id("kotlin-parcelize")
5 }
6
7 android {
8     namespace = "com.example.modul3"
9     compileSdk = 34
10 }
```

```
11     defaultConfig {
12         applicationId = "com.example.modul3"
13         minSdk = 30
14         targetSdk = 34
15         versionCode = 1
16         versionName = "1.0"
17
18         testInstrumentationRunner =
19 "androidx.test.runner.AndroidJUnitRunner"
20     }
21
22     buildTypes {
23         release {
24             isMinifyEnabled = false
25             proguardFiles(
26                 getDefaultProguardFile("proguard-android-
27 optimize.txt"),
28                 "proguard-rules.pro"
29             )
30         }
31     }
32
33     compileOptions {
34         sourceCompatibility = JavaVersion.VERSION_17
35         targetCompatibility = JavaVersion.VERSION_17
36     }
37
38     kotlinOptions {
39         jvmTarget = "17"
40     }
41
42     buildFeatures {
43         viewBinding = true
44     }
```

```
45 }
46
47 dependencies {
48     implementation("androidx.core:core-ktx:1.12.0")
49     implementation("org.jetbrains.kotlin:kotlin-stdlib:1.9.0")
50     implementation("androidx.appcompat:appcompat:1.6.1")
51     implementation("com.google.android.material:material:1.11.0")
52
53     implementation("androidx.constraintlayout:constraintlayout:2.1.4")
54     implementation("androidx.recyclerview:recyclerview:1.3.2")
55     implementation("androidx.cardview:cardview:1.0.0")
56     implementation("androidx.lifecycle:lifecycle-runtime-
57 ktx:2.6.2")
58     implementation("androidx.activity:activity-ktx:1.8.2")
59
60     testImplementation("junit:junit:4.13.2")
61     androidTestImplementation("androidx.test.ext:junit:1.1.5")
62     androidTestImplementation("androidx.test.espresso:espresso-
63 core:3.5.1")
64 }
```

B. Output Program



2:34

Gunung Bromo
Jawa Timur
Gunung Bromo adalah gunung api aktif yang terkenal secara internasional karena lanskapnya yang ikonik dan mudah diakses dengan ketinggian 2.329 mdpl. Gunung ini merupakan bagian dari kaldera Tengger yang ...

[Link](#) [Detail](#)

Gunung Kerinci
Jambi, di Taman Nasional Kerinci Seblat
Gunung Kerinci adalah puncak tertinggi di Pulau Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan total ketinggian 3.805 mdpl. Gunung ini berdiri megah di tengah kawasan konservasi Taman Nasional...

[Link](#) [Detail](#)

2:34

Gunung Merbabu
Jawa Tengah
Gunung Merbabu adalah gunung dengan ketinggian 3.145 meter di atas permukaan laut (mdpl) yang bertipe stratovolcano yang sudah tidak aktif, berdampingan erat dengan Gunung Merapi di sebelah selatannya. Nam...

[Link](#) [Detail](#)

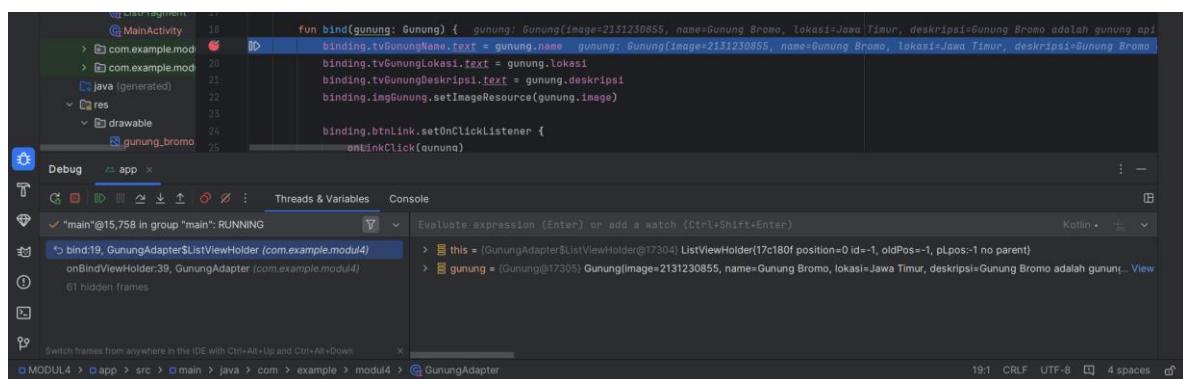
Gunung Rinjani
Lombok, Nusa Tenggara Barat
Gunung Rinjani merupakan gunung berapi tertinggi kedua di Indonesia dan menjadi ikon Pulau Lombok dengan ketinggian 3.726 mdpl. Gunung ini termasuk dalam Taman Nasional Gunung Rinjani yang luasnya me...

[Link](#) [Detail](#)

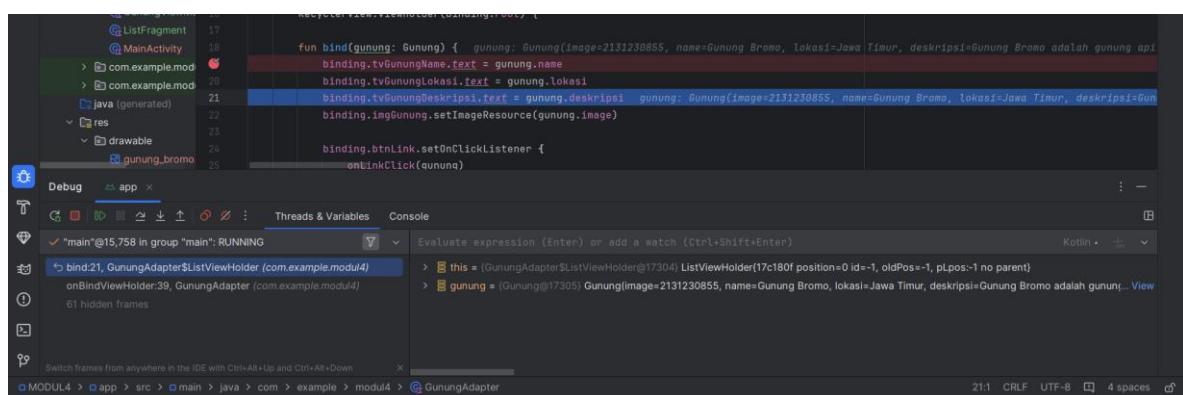




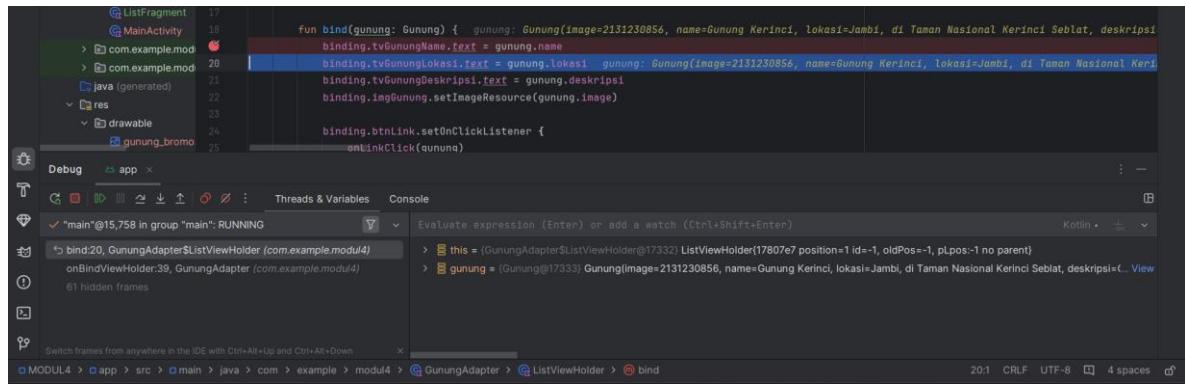
Gambar 4. Screenshot Item List Gunung Hasil Jawaban Soal 1 Modul 4



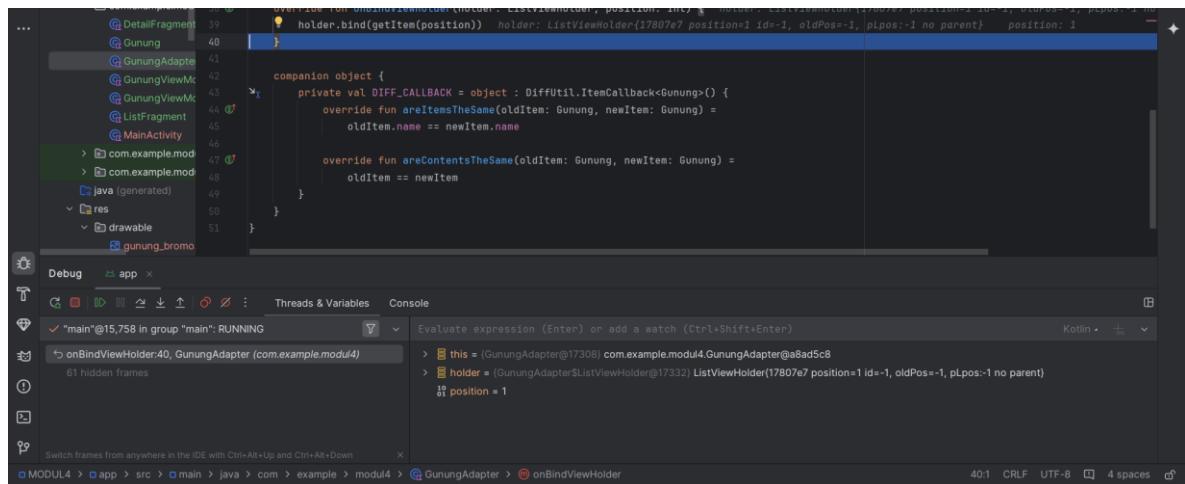
Gambar 5. Debugger



Gambar 6. Step Over (F8)



Gambar 7. Step Into (F7)



Gambar 8. Step Out (Shift+F8)

C. Pembahasan

app\src\main\java\com\example\modul3

15. DetailFragment.kt:

Pada baris [1] terdapat fungsi package com.example.modul4 yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.modul4. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] hingga [8] terdapat fungsi import yang digunakan untuk mengimpor berbagai komponen yang dibutuhkan dalam pengembangan aplikasi Android, diantaranya untuk menampilkan detail informasi berupa nama, lokasi, deskripsi, dan gambar, yang dikirim melalui Bundle arguments, ViewBinding (DetailFragmentBinding)

untuk mengakses view secara aman, dan mengatur ulang binding saat view dihancurkan guna mencegah memory leak. Pada baris [10] digunakan untuk mendeklarasikan sebuah kelas bernama DetailFragment yang merupakan subclass dari Fragment dalam Android. Pada baris [12] dan [13] terdapat fungsi private var _binding: DetailFragmentBinding? = null dan private val binding get() = _binding!! digunakan untuk mengelola ViewBinding dalam sebuah fragment secara aman. Variabel _binding bertipe nullable (DetailFragmentBinding?) berfungsi untuk menyimpan objek binding yang menghubungkan class Kotlin dengan layout XML (detail_fragment.xml). Binding ini diinisialisasi saat onCreateView() dan dihapus (null) pada onDestroyView() guna mencegah kebocoran memori karena siklus hidup View pada fragment bisa berbeda dari fragment itu sendiri. Sementara itu, properti binding merupakan versi non-null dari _binding, yang memanfaatkan operator !! untuk memastikan bahwa binding hanya digunakan ketika sudah pasti tidak null. Dengan pendekatan ini, kita dapat mengakses elemen-elemen view secara aman dan efisien tanpa perlu memanggil findViewById, serta tetap menjaga praktik pemrograman yang sesuai dengan lifecycle fragment. Pada baris [15] hingga [20] terdapat fungsi yang digunakan untuk membuat dan mengembalikan tampilan (view) dari fragment saat fragment sedang dibuat. Parameter inflater digunakan untuk "meng-inflate" layout XML menjadi objek view, sedangkan container adalah parent view tempat fragment akan ditempelkan, dan savedInstanceState menyimpan data keadaan sebelumnya jika ada. Di dalam metode ini, digunakan DetailFragmentBinding.inflate(...) untuk menghubungkan layout detail_fragment.xml dengan objek binding _binding. Proses inflate ini membuat layout XML bisa diakses melalui properti binding, sehingga memudahkan dalam pengelolaan tampilan secara efisien tanpa perlu findViewById. Setelah di-inflate, biasanya method ini akan mengembalikan binding.root sebagai tampilan utama fragment (meskipun belum terlihat di sini). Pendekatan ini memastikan integrasi yang aman antara tampilan dan logika dalam fragment, serta mengikuti praktik modern pengembangan Android menggunakan ViewBinding.

Pada baris [22] hingga [25] terdapat fungsi kode yang digunakan untuk **mengambil data yang dikirimkan ke fragment melalui Bundle arguments**.

Masing-masing baris berfungsi untuk mengambil data berdasarkan key yang telah ditentukan, seperti "EXTRA_PHOTO" untuk gambar (tipe Int), "EXTRA_NAME" untuk nama (tipe String), "EXTRA_LOKASI" untuk lokasi, dan "EXTRA_DESKRIPSI" untuk deskripsi. Metode getInt() dan getString() dipanggil secara aman menggunakan operator ?, yang artinya data hanya akan diambil jika arguments tidak bernilai null. Biasanya, data ini dikirim dari fragment atau activity sebelumnya saat navigasi ke DetailFragment, dan nantinya akan ditampilkan ke dalam elemen UI fragment. Pada baris [28] hingga [25] terdapat fungsi kode yang menampilkan data yang diterima dari arguments ke dalam elemen-elemen UI pada fragment menggunakan ViewBinding. Pertama, nilai variabel name, lokasi, dan deskripsi yang telah diambil dari arguments diset ke dalam TextView yang sesuai, seperti tvName, tvLokasi, dan tvDeskripsi. Hal ini memungkinkan fragment untuk menampilkan informasi seperti nama, lokasi, dan deskripsi pada tampilan UI. Selanjutnya, jika ada nilai untuk gambar (yang disimpan dalam variabel image), kode menggunakan let untuk memastikan bahwa nilai tersebut tidak null sebelum mengatur gambar pada ImageView (imgPoster) dengan menggunakan setImageResource(). Terakhir, metode return binding.root mengembalikan root view dari layout yang telah di-bind, yaitu tampilan fragment yang sudah lengkap dengan data yang ditampilkan. Pendekatan ini menggabungkan ViewBinding untuk mengakses tampilan UI dengan cara yang lebih aman dan efisien, menghindari penggunaan findViewById yang rentan terhadap kesalahan dan meningkatkan keterbacaan kode.

Pada baris [38] hingga [40] terdapat Metode onDestroyView(), bagian dari siklus hidup fragment yang dipanggil ketika tampilan fragment akan dihancurkan atau saat fragment tidak lagi ditampilkan di layar. Dalam metode ini, super.onDestroyView() dipanggil untuk memastikan bahwa proses penghancuran tampilan fragment dilakukan dengan benar oleh sistem. Setelah itu, _binding = null digunakan untuk **menghapus referensi ke objek binding** dan mencegah terjadinya **memory leak**. Hal ini penting karena meskipun fragment dihancurkan, tampilan (view) yang digunakan oleh fragment bisa tetap ada di memori jika referensinya tidak dihapus. Dengan menyetel _binding ke null, kita memastikan bahwa objek tersebut tidak lagi mengacu pada tampilan yang sudah dihancurkan, sehingga sumber daya

dapat dibebaskan dan memori tidak terbuang sia-sia. Pendekatan ini adalah best practice untuk mengelola siklus hidup fragment dan mencegah masalah terkait memori dalam aplikasi Android.

16. Gunung.kt:

Pada baris [1] tersebut mendefinisikan sebuah kelas data bernama Gunung yang mengimplementasikan interface Parcelable. Pada baris [7] hingga [12] menampilkan lima property yakni image (menyimpan ID sumber daya gambar), name (nama gunung), lokasi (lokasi gunung), deskripsi (deskripsi gunung), dan link (URL terkait gunung). Dengan menambahkan anotasi `@Parcelize`, kelas Gunung secara otomatis mendapatkan implementasi dari metode Parcelable, yang memungkinkan objek dari kelas ini untuk diserialisasi dan dipassing antar komponen dalam aplikasi Android, seperti Activity atau Fragment, melalui Intent atau Bundle. Anotasi ini menyederhanakan proses karena Android akan menangani semua detail terkait serialisasi dan deserialisasi objek, sehingga developer tidak perlu menulis kode secara manual. Keuntungan utama dari menggunakan Parcelable adalah efisiensi kinerja dalam mengirim data antar komponen di aplikasi, yang lebih cepat dibandingkan menggunakan Serializable. Dengan pendekatan ini, objek Gunung dapat dengan mudah dipindahkan dan dipertukarkan antar aktivitas atau fragment dalam aplikasi.

17. GunungAdapter.kt

Kode di atas merupakan implementasi kelas GunungAdapter di Android yang digunakan untuk menampilkan daftar objek Gunung dalam komponen RecyclerView dengan menggunakan pendekatan ListAdapter. Setiap item daftar akan ditampilkan dengan data nama gunung, lokasi, deskripsi, dan gambar, serta memiliki dua tombol aksi: satu untuk membuka link dan satu lagi untuk melihat detail. Pada baris [1] hingga [8] terdapat package com.example.modul4 menyatakan bahwa file ini merupakan bagian dari paket com.example.modul4. Kemudian, sejumlah library penting diimpor seperti LayoutInflater, ViewGroup, DiffUtil, ListAdapter, RecyclerView, dan ItemGunungBinding. LayoutInflater digunakan untuk mengubah

layout XML menjadi objek View, ViewGroup adalah parent dari tampilan item, DiffUtil berguna untuk membandingkan data saat daftar berubah, dan ListAdapter adalah turunan dari RecyclerView.Adapter yang menyediakan fitur otomatis dalam membandingkan dan memperbarui daftar data menggunakan DiffUtil.

Pada baris [10] hingga [33] terdapat kelas GunungAdapter adalah turunan dari ListAdapter dengan tipe data Gunung dan ListViewHolder. Adapter ini memiliki dua parameter fungsi yaitu onLinkClick dan onDetailClick, yang didefinisikan sebagai lambda function yang menerima objek Gunung sebagai argumen. Fungsi-fungsi ini digunakan untuk menangani klik pada tombol link dan detail. Di dalam kelas ini terdapat inner class ListViewHolder, yang bertugas untuk memegang referensi view dari item layout (ItemGunungBinding) dan mengatur data ke dalam view tersebut. Dalam metode bind, data gunung ditampilkan ke dalam TextView dan ImageView yang sesuai. Metode ini juga menetapkan aksi klik pada dua tombol (btnLink dan btnDetail) dengan cara memanggil lambda function yang dikirimkan melalui constructor GunungAdapter. Pada baris [35] terdapat fungsi onCreateViewHolder bertugas untuk membuat view holder baru saat diperlukan. Di sini, layout item_gunung.xml di-inflate menjadi objek View menggunakan ItemGunungBinding.inflate, dan kemudian dibungkus dalam instance ListViewHolder. Sementara itu, onBindViewHolder dipanggil saat sistem ingin menampilkan data pada posisi tertentu dalam daftar. Fungsi ini cukup memanggil bind pada holder yang sesuai, dan meneruskan data yang diambil menggunakan getItem(position).

Terakhir, pada baris [48] dalam blok companion object, terdapat objek statis DIFF_CALLBACK yang digunakan oleh ListAdapter untuk mengetahui perbedaan antara dua item dalam daftar. Metode areItemsTheSame digunakan untuk membandingkan apakah dua item adalah entitas yang sama (dalam hal ini dibandingkan berdasarkan nama gunung), sedangkan areContentsTheSame memeriksa apakah konten kedua item benar-benar sama. DiffUtil secara otomatis akan menangani animasi dan efisiensi pembaruan daftar saat data berubah. Sehingga, GunungAdapter ini memungkinkan pengelolaan dan penampilan data daftar gunung

secara efisien dan dinamis dalam RecyclerView, lengkap dengan interaksi pengguna melalui tombol-tombol di setiap item.

18. GunungViewModel

Kode kelas ini merupakan GunungViewModel yang berada dalam paket com.example.modul4.viewmodel. Kelas ini merupakan turunan dari ViewModel dan bertanggung jawab untuk menyimpan serta mengelola data terkait objek Gunung secara berkelanjutan di dalam aplikasi Android berbasis arsitektur MVVM (Model-View-ViewModel). ViewModel ini membantu menjaga data tetap tersedia saat terjadi perubahan konfigurasi seperti rotasi layar, dan menyediakan cara yang reaktif untuk memperbarui tampilan dengan menggunakan StateFlow. Pada baris [11] hingga [21] terdapat deklarasi awal menunjukkan bahwa GunungViewModel menerima parameter initialList, yaitu sebuah daftar objek Gunung yang akan dijadikan data awal. Kemudian, terdapat dua properti utama yang menggunakan MutableStateFlow, yaitu _gunungList dan _selectedGunung. Properti _gunungList adalah daftar seluruh gunung, dan dibungkus dalam properti publik gunungList bertipe StateFlow agar hanya dapat dibaca dari luar ViewModel. Ini digunakan untuk memantau perubahan daftar gunung secara reaktif dari layer UI. Hal serupa juga berlaku untuk _selectedGunung, yang menyimpan data gunung yang dipilih saat pengguna menekan tombol detail—dapat dipantau dari UI melalui selectedGunung.

Pada baris [23] hingga [29] terdapat blok init, kode yang berada dalam viewModelScope.launch akan dijalankan saat ViewModel diinisialisasi. viewModelScope memastikan coroutine berjalan dalam siklus hidup ViewModel dan dibatalkan secara otomatis ketika ViewModel dihancurkan. Di dalam coroutine tersebut, nilai _gunungList diisi dengan initialList, dan dicetak log debug menggunakan Log.d untuk memberi informasi bahwa data gunung berhasil dimuat, lengkap dengan jumlah itemnya. Pada baris [31] hingga [35] terdapat fungsi onDetailClicked(gunung: Gunung) digunakan saat tombol detail diklik oleh pengguna. Fungsi ini akan mengubah nilai _selectedGunung menjadi gunung yang diklik, sehingga UI yang mengamati selectedGunung dapat menampilkan detailnya. Fungsi ini juga mencetak log untuk tujuan debugging agar developer mengetahui gunung mana yang sedang dilihat detailnya. Sementara itu, pada baris [37] hingga

[41] terdapat fungsi `onLinkClicked(gunung: Gunung)` hanya mencetak pesan ke log ketika pengguna menekan tombol yang akan membuka link (biasanya menggunakan intent eksplisit ke browser atau aplikasi lain). Walaupun tidak mengubah state seperti `onDetailClicked`, fungsi ini tetap penting untuk mencatat interaksi pengguna dan bisa dikembangkan untuk memicu aksi nyata seperti membuka URL. Sehingga, kelas `GunungViewModel` ini menyediakan pengelolaan data dan logika interaksi yang terpisah dari tampilan (UI), menggunakan `StateFlow` untuk mengaktifkan pola observasi data yang reaktif, serta menggunakan coroutine untuk memastikan operasi data dilakukan secara efisien dan aman dari blocking thread utama. Ini adalah implementasi `ViewModel` yang baik dalam konteks arsitektur Android modern.

19. GunungViewModelFactory

Kode di atas merupakan implementasi dari `GunungViewModelFactory`, yaitu sebuah kelas yang mengimplementasikan interface `ViewModelProvider.Factory` untuk membuat instance `GunungViewModel` dengan argumen khusus. Kelas ini berada dalam paket `com.example.modul4.viewmodel` dan berfungsi sebagai *factory* atau pabrik pembuat `ViewModel` secara manual, terutama saat `ViewModel` membutuhkan parameter di konstruktor, seperti daftar gunung (`gunungList`) dalam kasus ini.

Pada baris [7] hingga [19] terdapat fungsi kelas `GunungViewModelFactory` memiliki satu properti yaitu `gunungList`, yang bertipe `List<Gunung>`. Properti ini diterima dari luar dan nantinya akan diteruskan ke konstruktor `GunungViewModel`. Karena `ViewModelProvider` secara default hanya bisa membuat `ViewModel` tanpa parameter (menggunakan konstruktor kosong), maka kita membutuhkan kelas factory ini untuk menyuntikkan data awal secara eksplisit ke dalam `ViewModel`. Fungsi `create` adalah override dari fungsi yang didefinisikan di interface `ViewModelProvider.Factory`. Fungsi ini menerima parameter `modelClass`, yaitu kelas `ViewModel` yang ingin dibuat. Di dalam fungsi ini, dilakukan pemeriksaan apakah `modelClass` merupakan kelas `GunungViewModel` atau subclass-nya, menggunakan fungsi `isAssignableFrom`. Jika benar, maka dibuat dan dikembalikan instance `GunungViewModel`, dengan menyertakan `gunungList` sebagai parameter. Karena tipe pengembalian bersifat generik (`T : ViewModel`), maka dibutuhkan

casting secara eksplisit dengan anotasi `@Suppress("UNCHECKED_CAST")` untuk menghindari peringatan kompilator. Jika `modelClass` bukan `GunungViewModel`, maka akan dilemparkan exception `IllegalArgumentException` dengan pesan bahwa kelas `ViewModel` tidak dikenali. Ini penting untuk mencegah aplikasi membuat `ViewModel` yang salah dan menimbulkan error yang sulit dilacak.

20. ListFragment.kt

Kode `ListFragment` pada kelas ini merupakan bagian dari aplikasi Android yang menggunakan arsitektur MVVM dan menampilkan daftar data gunung dalam bentuk list menggunakan `RecyclerView`. Kelas ini merupakan turunan dari `Fragment`, dan berfungsi sebagai antarmuka utama tempat data gunung ditampilkan serta interaksi awal seperti membuka link atau melihat detail gunung dilakukan. Pada baris [21] hingga [24] terdapat bagian awal kelas, terdapat properti `_binding` yang digunakan untuk mengakses komponen UI dari layout `list_fragment.xml` melalui mekanisme **View Binding**. Properti binding dibuat sebagai akses aman (non-nullable) ke `_binding` untuk digunakan selama Fragment masih aktif. Kemudian dideklarasikan `viewModel` bertipe `GunungViewModel`, yang bertanggung jawab mengelola data dan logika presentasi untuk fragment ini.

Dalam baris [26] hingga [36] terdapat metode `onCreateView`, layout dari fragment di-*inflate* menggunakan `ListFragmentBinding`. Selanjutnya, data gunung dimuat dari resource melalui fungsi `getListGunung()`, yang menggabungkan array `image`, `name`, `lokasi`, `deskripsi`, dan link dari file `res/values/strings.xml` dan `arrays.xml`. Data ini kemudian diberikan ke `GunungViewModel` melalui `GunungViewModelFactory`, karena `ViewModel` membutuhkan konstruktor dengan parameter (tidak bisa dibuat secara langsung). Pada baris [38] hingga baris [70] terdapat metode `setupRecyclerView()` dipanggil. Di sini dibuat objek `GunungAdapter` dengan dua lambda sebagai parameter: `onLinkClick` dan `onDetailClick`. Saat pengguna mengklik tombol link, `viewModel.onLinkClicked()` dipanggil dan `Intent.ACTION_VIEW` digunakan untuk membuka link di browser. Saat tombol detail diklik, `ViewModel` memperbarui `selectedGunung`, lalu navigasi ke `DetailFragment` dilakukan dengan mengirim data gunung lewat `Bundle` (menggunakan `.putInt`, `putString`, dll). Proses navigasi ini menggunakan

FragmentTransaction untuk mengganti konten frame_container dengan fragment detail dan menambahkan transaksi ke backstack agar pengguna bisa kembali. Pada baris [71] hingga [81] terdapat RecyclerView yang dikonfigurasi menggunakan LinearLayoutManager secara vertikal. Adapter ditetapkan, dan metode setHasFixedSize(true) digunakan agar ukuran item tidak berubah-ubah, meningkatkan performa. Data gunung dari ViewModel dikoleksi menggunakan coroutine pada lifecycleScope.launchWhenStarted, sehingga setiap perubahan data di gunungList akan langsung ditampilkan oleh adapter dengan submitList. Pada baris [83] hingga [92] terdapat metode observeViewModel() digunakan untuk memantau selectedGunung yang dipilih. Meskipun tidak mengubah UI secara eksplisit di sini, log dicetak untuk membantu debugging dan pelacakan navigasi detail.

Pada baris [94] hingga baris [115] terdapat fungsi getListGunung() yang bertanggung jawab untuk membuat daftar Gunung dengan membaca berbagai data dari resource. obtainTypedArray() digunakan untuk mengambil resource ID gambar, dan setelah digunakan, array tersebut *direcycle* untuk menghindari memory leak. Terakhir, pada baris [117] hingga [121] terdapat fungsi onDestroyView() memastikan binding dibersihkan ketika view fragment dihancurkan, sesuai praktik aman dalam penggunaan View Binding di Fragment agar tidak terjadi kebocoran memori. Sehingga, ListFragment adalah implementasi yang lengkap dan modular untuk menampilkan dan menangani interaksi dengan daftar data gunung, menggunakan prinsip arsitektur MVVM, View Binding, dan coroutine berbasis StateFlow untuk data yang reaktif.

21. MainActivity.kt

Pada baris [1] terdapat fungsi package com.example.modul3 yang digunakan untuk mendeklarasikan bahwa file tersebut termasuk dalam paket bernama com.example.modul3. (package) berfungsi untuk mengelompokkan kelas-kelas yang saling berhubungan agar kode lebih terstruktur dan mudah dikelola. Pada baris [3] dan [4] terdapat beberapa import, baris import android.os.Bundle digunakan untuk mengimpor kelas Bundle, yaitu objek yang berfungsi menyimpan dan mengelola data sementara yang digunakan dalam pengiriman data antar komponen Android, seperti

saat menyimpan state activity. Sedangkan import androidx.appcompat.app.AppCompatActivity digunakan untuk mengimpor AppCompatActivity, yaitu kelas dasar untuk activity yang memberikan dukungan kompatibilitas ke versi Android lama dan memungkinkan penggunaan fitur modern seperti Toolbar dan Fragment secara lebih konsisten. Keduanya biasanya digunakan saat membuat kelas Activity di Android.

Pada baris [7] hingga [15] terdapat fungsi kelas MainActivity mewarisi AppCompatActivity, dan di dalam metode onCreate() dilakukan inisialisasi tampilan utama menggunakan setContentView(R.layout.activity_main). Selanjutnya, supportFragmentManager digunakan untuk mengelola fragment. Sebuah instance ListFragment dibuat dan disiapkan untuk ditambahkan ke MainActivity. Kemudian, baris findFragmentByTag(...) digunakan untuk memeriksa apakah fragment dengan tag nama kelas ListFragment sudah ada sebelumnya, yang berguna untuk menghindari penambahan fragment secara berulang saat konfigurasi ulang seperti rotasi layar. Pada baris [17] hingga [22] terdapat fungsi yang berguna untuk memeriksa apakah fragment dengan tag ListFragment belum ada pada fragmentManager. Jika belum (fragment !is ListFragment), maka dilakukan transaksi fragment menggunakan beginTransaction(). Melalui add(), fragment listFragment ditambahkan ke dalam R.id.frame_container, yaitu sebuah ViewGroup di layout activity_main.xml yang menjadi wadah tampilan fragment. Tag yang digunakan adalah nama kelas ListFragment agar fragment ini bisa dikenali kembali di kemudian waktu. Akhirnya, commit() digunakan untuk menyelesaikan dan mengeksekusi transaksi fragment tersebut secara efektif. Pendekatan ini umum dalam aplikasi Android berbasis fragment untuk mengatur tampilan secara modular.

app\src\main\res\layout

22. activity_main.xml

Kode pada kelas ini merupakan layout activity_main.xml yang digunakan sebagai tampilan utama dari MainActivity dalam aplikasi Android. Layout ini menggunakan FrameLayout sebagai elemen root, yang berfungsi sebagai wadah (container) untuk menampung satu atau beberapa tampilan (biasanya Fragment) di

dalamnya. Atribut `android:id="@+id/frame_container"` memberikan ID pada FrameLayout, sehingga dapat diakses dan dimanipulasi dari kode Kotlin, misalnya saat menambahkan ListFragment ke dalamnya. Ukuran layout diatur memenuhi layar dengan `match_parent` untuk lebar dan tinggi. Namespace `tools:context` menunjukkan bahwa layout ini akan digunakan oleh MainActivity, dan digunakan oleh Android Studio untuk tujuan preview. FrameLayout dipilih karena cocok untuk menampilkan satu Fragment pada satu waktu secara bertumpuk.

23. `detail_fragment.xml`

Kode xml ini merupakan layout untuk `detail_fragment.xml` yang digunakan oleh `DetailFragment.kt`. Pada baris [1] hingga [10] terdapat layout yang menggunakan elemen ScrollView sebagai root, yang memungkinkan konten di dalamnya untuk dapat digulir secara vertikal jika melebihi tinggi layar. Atribut `layout_width` dan `layout_height` disetel ke `match_parent`, sehingga ScrollView akan mengisi seluruh layar. Warna latar belakang diatur ke `#B6D6F1`, memberikan nuansa biru muda yang lembut. Padding sebesar `16dp` ditambahkan ke seluruh sisi untuk memberikan ruang antara konten dan tepi layar. Namespace `tools:context` menunjukkan bahwa layout ini milik `DetailFragment`, berguna untuk preview di Android Studio. Umumnya, di dalam ScrollView akan ada LinearLayout vertikal yang berisi elemen-elemen UI seperti gambar, teks, dan tombol.

Pada baris [12] hingga [14] terdapat fungsi yang mendefinisikan sebuah `ConstraintLayout` yang merupakan wadah tata letak fleksibel dalam Android. `ConstraintLayout` digunakan di dalam ScrollView untuk menyusun elemen-elemen UI secara fleksibel dengan menggunakan constraint atau batasan antar elemen, tanpa harus membuat hierarki layout yang dalam. Atribut `layout_width="match_parent"` berarti layout ini akan mengambil seluruh lebar dari parent-nya (dalam hal ini, ScrollView), sedangkan `layout_height="wrap_content"` berarti tinggi layout akan mengikuti tinggi konten di dalamnya. `ConstraintLayout` sering dipilih karena efisien dalam kinerja dan memungkinkan desain yang kompleks tanpa banyak nesting layout. Di dalamnya biasanya terdapat berbagai elemen UI (seperti `ImageView`, `TextView`, `Button`) yang diatur posisinya relatif terhadap satu sama lain. Pada baris

[16] hingga [23] terdapat baris komponen ImageView di dalam ConstraintLayout yang digunakan untuk menampilkan gambar (dalam konteks ini, kemungkinan gambar gunung). android:id="@+id/imgPoster" memberikan ID unik agar dapat diakses di Kotlin melalui ViewBinding atau findViewById. Atribut layout_width="0dp" digunakan bersama constraint Start dan End, yang berarti lebarnya akan disesuaikan dengan lebar antara batas kiri dan kanan parent-nya. layout_height="300dp" menetapkan tinggi tetap sebesar 300dp. Atribut contentDescription="Foto Gunung" penting untuk aksesibilitas, memberi tahu pengguna pembaca layar bahwa ini adalah gambar gunung. Constraint Top_toTopOf="parent", Start_toStartOf="parent", dan End_toEndOf="parent" memastikan gambar berada di bagian atas dan terpusat secara horizontal dalam layout. Pendekatan ini umum digunakan dalam desain modern Android agar UI responsif dan konsisten di berbagai ukuran layar.

Pada baris [25] hingga [37] terdapat fungsi TextView di atas digunakan untuk menampilkan nama gunung dengan tampilan mencolok di bawah gambar. ID @+id/tvName memungkinkan TextView ini diakses melalui kode Kotlin (misalnya dengan ViewBinding). Lebarnya diset 0dp karena menggunakan constraint horizontal (Start dan End) agar mengisi ruang dari kiri ke kanan layout induk. layout_height="wrap_content" membuat tingginya menyesuaikan isi teks. Atribut gravity="center" memusatkan teks dalam TextView, sedangkan textSize="35sp" dan textStyle="bold" membuat teks terlihat besar dan tebal, ideal untuk judul. Warna teks hitam ditentukan dengan textColor="@android:color/black" agar kontras. tools:text="Gunung Semeru" hanya digunakan untuk preview di Android Studio dan tidak muncul saat runtime. Constraint Top_toBottomOf="@id/imgPoster" menempatkan TextView tepat di bawah gambar, sedangkan Start dan End dikaitkan ke parent untuk membuatnya berada di tengah secara horizontal. Layout ini mendukung tampilan yang bersih, responsif, dan estetis. Pada baris [39] hingga [49] terdapat fungsi TextView ini digunakan untuk menampilkan lokasi gunung dan ditempatkan tepat di bawah nama gunung. Lebarnya diatur 0dp agar mengisi ruang horizontal antara batas kiri dan kanan parent layout (menggunakan constraintStart dan constraintEnd). Tingginya otomatis menyesuaikan isi teks karena wrap_content.

Teks ditampilkan di tengah menggunakan gravity="center" dan ukuran font-nya 22sp, cocok untuk informasi tambahan. Atribut tools:text="Lokasi: Jawa Timur" hanya digunakan sebagai contoh pratinjau di Android Studio, bukan untuk tampilan saat aplikasi dijalankan. layout_marginTop="8dp" memberi jarak ke atas agar tidak terlalu rapat dengan teks nama gunung. Constraint Top_toBottomOf="@+id/tvName" menempatkan TextView ini tepat di bawah elemen nama, menjaga susunan konten yang rapi dan konsisten.

Pada baris [51] hingga [66] terdapat fungsi xml yang digunakan untuk tampilan halaman detail untuk informasi gunung dalam aplikasi Android. Struktur utamanya dibungkus oleh ScrollView, yang memungkinkan seluruh konten bisa digulir secara vertikal jika melebihi tinggi layar. Di dalam ScrollView, terdapat ConstraintLayout yang berfungsi sebagai wadah fleksibel untuk menyusun komponen UI dengan posisi yang saling terikat. Komponen yang digunakan terdiri dari ImageView untuk menampilkan gambar gunung di bagian atas, kemudian diikuti oleh tiga TextView yang masing-masing menampilkan nama gunung, lokasi, dan deskripsi secara terstruktur. Nama gunung ditampilkan dengan ukuran huruf besar dan tebal agar menjadi fokus utama, sedangkan lokasi ditampilkan dengan ukuran sedang. Deskripsi ditata menggunakan mode perataan antar kata (justificationMode="inter_word") agar terlihat lebih rapi dan nyaman dibaca. Seluruh elemen disusun secara responsif dan simetris di tengah layar, menciptakan antarmuka yang bersih dan informatif bagi pengguna.

24. Item_gunung.xml

Kode xml ini merupakan awal dari layout item yang menggunakan CardView sebagai kontainer utama. Pada baris [1] hingga [10] terdapat fungsi CardView ini berfungsi untuk membungkus satu item data (misalnya data gunung) dengan tampilan yang rapi dan memiliki efek elevasi (bayangan) serta sudut melengkung. Properti layout_width diset ke match_parent agar lebar kartu menyesuaikan dengan lebar parent, sedangkan layout_height diset ke wrap_content agar menyesuaikan tinggi konten di dalamnya. Margin luar diberikan sebesar 8dp agar antar item tidak saling menempel. Efek bayangan diatur melalui cardElevation sebesar 4dp dan

cardCornerRadius sebesar 8dp untuk memberi kesan visual modern dan menarik. Biasanya, elemen UI seperti gambar dan teks akan ditempatkan di dalam CardView ini untuk menampilkan informasi setiap item dalam RecyclerView. Pada baris [12] hingga [15] terdapat fungsi ConstraintLayout digunakan sebagai layout utama di dalam CardView. ConstraintLayout dipilih karena fleksibilitasnya dalam mengatur posisi elemen UI dengan constraint antar komponen. Lebarnya diset match_parent agar memenuhi lebar CardView, sedangkan tingginya wrap_content, artinya akan menyesuaikan tinggi konten di dalamnya. Properti padding="25dp" memberikan ruang di dalam layout agar isi seperti teks dan gambar tidak menempel langsung ke tepi kartu, sehingga tampilan lebih rapi dan nyaman dilihat.

Pada baris [17] hingga [27] terdapat fungsi ImageView dengan ID imgGunung yang digunakan untuk menampilkan gambar gunung dalam setiap item list. Ukuran gambar disetel sebesar 120dp lebar dan 160dp tinggi, dengan scaleType="centerCrop" agar gambar memenuhi ruang tanpa mengubah rasio secara tidak proporsional. Gambar ini ditempatkan di sisi kiri item dengan batas kanan (end) yang terhubung ke komponen linearLayoutText, dan diberi margin end sebesar 12dp agar tidak menempel langsung. Posisi atas dan kiri dikaitkan ke parent untuk memastikan gambar berada di bagian atas dan kiri dari kartu. Pada baris [29] hingga [38] terdapat fungsi LinearLayout dengan ID linearLayoutText ini digunakan untuk menampung elemen teks seperti nama gunung, lokasi, dan deskripsi secara vertikal di sebelah kanan gambar. Lebarnya disetel 0dp agar mengikuti aturan ConstraintLayout, dan tingginya wrap_content menyesuaikan isi. Komponen ini dikaitkan dengan sisi kanan (end) dari imgGunung, sisi atas parent, dan sisi kanan parent, sehingga menempati sisa ruang horizontal di samping gambar. Margin atas sebesar 8dp memberikan jarak dari atas, dan meskipun terdapat atribut layout_weight, atribut ini tidak berlaku di dalam ConstraintLayout sehingga bisa diabaikan atau dihapus.

Pada baris [40] hingga [47] terdapat fungsi TextView dengan ID tvGunungName berfungsi untuk menampilkan nama gunung pada setiap item dalam daftar. Komponen ini memiliki lebar dan tinggi yang disesuaikan secara otomatis

dengan isi teksnya, karena menggunakan atribut `wrap_content`. Ukuran teks diatur sebesar `24sp`, cukup besar untuk menarik perhatian pengguna sebagai judul utama pada tampilan kartu. Gaya teks dibuat tebal (`bold`) agar terlihat lebih mencolok dan menonjol, serta menggunakan warna hitam (`@android:color/black`) agar kontrasnya jelas dengan latar belakang. Untuk keperluan pratinjau di Android Studio, diberikan teks contoh "Gunung Semeru" melalui atribut `tools:text`, namun nilai ini tidak akan muncul saat aplikasi dijalankan. Desain ini membantu pengguna mengenali nama gunung secara cepat dan jelas pada tampilan daftar. Pada baris [49] hingga [56] terdapat fungsi `TextView` dengan ID `tvGunungLokasi` digunakan untuk menampilkan lokasi dari gunung yang sedang ditampilkan dalam item daftar. Elemen ini memiliki ukuran teks sebesar `14sp`, cukup kecil namun masih terbaca dengan jelas, memberikan informasi sekunder setelah nama gunung. Warna teks diatur menjadi abu-abu gelap dengan kode warna `#666666`, yang menandakan bahwa informasi ini bukan informasi utama, namun tetap penting. Gaya teks ditampilkan dalam bentuk tebal (`bold`) untuk menambah penekanan dan keterbacaan. Seperti biasa, atribut `tools:text` diisi dengan teks contoh "Lokasi: Jawa Timur" untuk memberikan pratinjau saat mendesain di Android Studio, namun tidak memengaruhi teks saat runtime. `TextView` ini membantu pengguna mengetahui dengan cepat lokasi geografis gunung dalam daftar.

Pada baris [58] hingga [69] terdapat fungsi `TextView` dengan ID `tvGunungDeskripsi` ini berfungsi untuk menampilkan deskripsi singkat mengenai gunung yang ditampilkan dalam setiap item daftar `RecyclerView`. Ukuran teksnya disesuaikan sebesar `14sp` agar tetap mudah dibaca, sementara warna hitam (`@android:color/black`) digunakan untuk memastikan keterbacaan di latar belakang terang. Untuk menjaga tata letak tetap rapi dan tidak memakan terlalu banyak ruang, deskripsi ini dibatasi maksimal dua baris menggunakan atribut `android:maxLines="2"`. Jika teks melebihi dua baris, maka akan dipotong dan ditandai dengan ellipsis (...) di akhir melalui `android:ellipsize="end"`. Penggunaan `tools:text` menampilkan contoh isi deskripsi saat proses desain, tanpa mempengaruhi tampilan aplikasi saat dijalankan.

Pada baris [71] hingga [82] terdapat fungsi LinearLayout ini digunakan untuk menampung dua tombol aksi (misalnya tombol "Detail" dan "Link") yang ditampilkan secara **horizontal** di bagian bawah informasi gunung pada setiap item RecyclerView. Layout ini memiliki lebar penuh (match_parent) dan tinggi menyesuaikan kontennya (wrap_content), dengan orientasi horizontal agar anak-anaknya (biasanya dua tombol) tersusun sejajar ke kanan. Atribut android:gravity="end" mengarahkan konten ke sisi kanan dari LinearLayout. Penambahan android:layout_marginTop="8dp" memberi jarak vertikal dari elemen di atasnya (teks), dan android:layout_marginStart="130dp" memberikan ruang kosong di sisi kiri agar posisi tombol tampak rata dengan bagian informasi teks di kanan gambar. Atribut weightSum="2" menandakan total bobot layout ini dibagi dua, sehingga masing-masing tombol di dalamnya dapat diberi layout_weight="1" untuk proporsi lebar yang seimbang. Constraint seperti app:layout_constraintStart_toStartOf="parent" dan app:layout_constraintTop_toBottomOf="@+id/linearLayoutText" menunjukkan bahwa LinearLayout ini masih berada dalam konteks ConstraintLayout, dan ditambatkan secara tepat di bawah elemen teks. Terakhir, pada baris [84] hingga [98] terdapat Dua tombol ini masing-masing diberi layout_width="0dp" dan layout_weight="1" agar keduanya membagi ruang secara merata di dalam LinearLayout yang memiliki weightSum="2". Tombol pertama, dengan id="@+id btnLink", memiliki teks "Link" dan layout_marginEnd="8dp" untuk memberi jarak horizontal antara dua tombol. Tombol kedua, btnDetail, memiliki teks "Detail". Keduanya memungkinkan pengguna untuk berinteraksi lebih lanjut: tombol "Detail" biasanya akan menampilkan informasi lengkap tentang gunung tersebut di fragment atau halaman baru, sementara tombol "Link" bisa diarahkan untuk membuka referensi eksternal, seperti artikel, website resmi, atau Google Maps.

25. list_fragment.xml

Kode pada kelas ini menampilkan layout dari kelas listfragment.kt. pada baris [1] hingga [8] terdapat fungsi constraintLayout agar elemen UI bisa diatur secara fleksibel dengan constraint antar komponen. Atribut tools:context=".ListFragment"

menunjukkan bahwa layout ini digunakan oleh kelas ListFragment. Dengan layout_width dan layout_height di-set ke match_parent, maka layout ini akan menempati seluruh ruang layar yang tersedia. Biasanya, di dalam layout ini akan ditambahkan sebuah RecyclerView untuk menampilkan daftar item seperti gunung, beserta komponen-komponen lain yang mungkin dibutuhkan seperti toolbar, filter, atau search bar. Pada baris [10] hingga [23] terdapat fungsi RecyclerView yang digunakan untuk menampilkan daftar gunung dalam aplikasi. Dengan ID rvGunung, komponen ini diatur agar menyesuaikan ukuran penuh layar dengan menggunakan constraint ke seluruh sisi parent layout. Atribut clipToPadding diset ke false agar konten bisa tetap terlihat saat di-scroll melewati padding. Latar belakang diberi warna biru muda (#94C2EB) dan diberi padding sebesar 16dp untuk memberikan jarak antara tepi layar dengan item di dalamnya. Atribut tools:listItem="@layout/item_gunung" hanya digunakan sebagai referensi desain di Android Studio, menunjukkan bahwa setiap item yang ditampilkan berasal dari layout item_gunung.xml.

app\src\main\res\values

26. colors.xml

Kode kelas ini berguna untuk mendefinisikan dua warna khusus untuk digunakan dalam aplikasi Android kamu. Warna blue_500 dengan nilai heksadesimal #2196F3 merupakan warna biru terang yang sering digunakan sebagai warna utama (primary color) dalam tema Material Design. Sedangkan blue_700 dengan nilai #1976D2 adalah versi yang lebih gelap dari warna biru tersebut, biasanya dipakai sebagai primaryDark atau untuk elemen-elemen yang membutuhkan aksen warna lebih kuat. Kedua warna ini dapat dipanggil di layout XML atau dalam kode Kotlin menggunakan resource ID seperti @color/blue_500 atau R.color.blue_700.

27. string.xml

Pada baris [1] hingga baris [9] terdapat fungsi Resource yang mendefinisikan nama aplikasi sebagai "MODUL 3" melalui elemen <string name="app_name">. Selain itu, juga terdapat sebuah array string bernama gunung_name yang memuat daftar lima gunung terkenal di Indonesia: Gunung Bromo, Gunung Kerinci, Gunung

Merapi, Gunung Merbabu, dan Gunung Rinjani. Array ini biasanya digunakan untuk menampilkan pilihan dalam komponen UI seperti Spinner, ListView, atau sebagai sumber data untuk list dinamis lainnya seperti RecyclerView. Pada baris [11] hingga [17] terdapat fungsi Array gunung_link berfungsi sebagai kumpulan tautan atau URL yang mengarah ke situs resmi masing-masing gunung yang ditampilkan dalam aplikasi. Setiap elemen dalam array ini berisi alamat website dari taman nasional atau pengelola resmi gunung seperti Bromo, Kerinci, Merapi, Merbabu, dan Rinjani. Tautan-tautan ini dimanfaatkan oleh aplikasi untuk memberikan akses langsung kepada pengguna terhadap informasi yang lebih lengkap dan terpercaya, seperti jalur pendakian, peraturan taman nasional, informasi perizinan, serta kegiatan konservasi. Biasanya, array ini akan dihubungkan dengan tombol di tampilan daftar atau detail gunung, sehingga ketika pengguna menekan tombol "Link", aplikasi akan membuka browser dan menampilkan halaman web sesuai URL yang bersangkutan. Pada baris [19] hingga [25] terdapat fungsi array gunung_lokasi yang digunakan untuk menyimpan informasi mengenai lokasi geografis dari setiap gunung yang ditampilkan dalam aplikasi. Setiap item dalam array ini berkorespondensi dengan item yang ada di array gunung_name, sehingga posisi masing-masing lokasi akan cocok dengan nama gunung yang bersangkutan. Misalnya, posisi pertama berisi "Jawa Timur" yang merupakan lokasi dari Gunung Bromo, sementara posisi kedua berisi "Jambi, di Taman Nasional Kerinci Seblat" untuk Gunung Kerinci, dan seterusnya. Informasi ini ditampilkan di antarmuka pengguna untuk memberi gambaran singkat kepada pengguna mengenai letak tiap gunung. Dengan menyusun data lokasi dalam string-array, developer dapat dengan mudah mengelola, menampilkan, dan memperbarui data lokasi secara efisien melalui adapter RecyclerView atau fragment detail dalam aplikasi Android.

Pada baris [27] hingga [125] yang diberikan mendefinisikan sebuah array string dengan nama gunung_deskripsi yang berisi beberapa item deskripsi tentang gunung. Array ini digunakan untuk menyimpan kumpulan teks atau informasi terkait gunung yang dapat diakses dalam aplikasi Android. Setiap elemen <item> di dalam array tersebut berisi deskripsi yang bisa digunakan untuk menampilkan informasi tentang gunung tertentu, seperti dalam tampilan daftar atau RecyclerView. Array ini

memungkinkan aplikasi untuk mengorganisir dan mengelola beberapa teks deskripsi dengan lebih terstruktur, sehingga memudahkan dalam mengakses dan menampilkan informasi terkait gunung secara dinamis. Pada baris [130] hingga [136] terdapat fungsi yang mendefinisikan sebuah array dengan nama gunung_image yang berisi beberapa item gambar yang terletak di direktori drawable aplikasi Android. Setiap elemen <item> di dalam array ini merujuk pada gambar yang berbeda, seperti @drawable/gunung_bromo, @drawable/gunung_kerinci, dan seterusnya. Gambar-gambar tersebut akan digunakan untuk mewakili berbagai gunung yang ada dalam aplikasi.

28. theme.xml

Kode kelas ini digunakan untuk mendefinisikan tema untuk aplikasi Android dengan menggunakan styles.xml. Tema ini dimulai dengan mendeklarasikan Base.Theme.Modul3, yang merupakan tema dasar yang diwarisi dari Theme.Material3.DayNight.NoActionBar, memberikan aplikasi dukungan mode terang dan gelap (DayNight) tanpa action bar. Di dalam tema dasar ini, beberapa atribut warna ditentukan, seperti colorPrimary yang menggunakan warna biru (@color/blue_500), colorOnPrimary yang menggunakan warna putih (@android:color/white) untuk teks atau ikon di atas elemen utama, serta colorPrimaryContainer yang menggunakan warna biru lebih gelap (@color/blue_700) untuk latar belakang elemen utama. Atribut colorOnPrimaryContainer juga diatur ke warna putih untuk teks di atas latar belakang tersebut. Selanjutnya, tema Theme.Modul3 mewarisi pengaturan dari Base.Theme.Modul3 dan dapat dimodifikasi lebih lanjut sesuai kebutuhan aplikasi. Dengan struktur ini, tema memastikan aplikasi memiliki konsistensi warna yang estetis dan mengikuti desain Material dengan mudah, sementara juga memberikan fleksibilitas dalam pengaturan tampilan aplikasi.

29. AndroidManifest.kt

Pada baris [3] hingga [6] terdapat fungsi import, diantaranya LayoutInflater digunakan untuk mengubah layout XML menjadi objek View yang dapat ditampilkan di UI, sementara ViewGroup digunakan untuk mendefinisikan grup tampilan tempat

item RecyclerView akan diletakkan. RecyclerView adalah komponen UI yang memungkinkan tampilan daftar item yang efisien dan dapat di-scroll. Kode ini juga mengimpor kelas ItemGunungBinding, yang dihasilkan otomatis oleh Android Studio ketika menggunakan ViewBinding. Dengan ViewBinding, kita bisa mengakses elemen-elemen UI dalam layout secara langsung tanpa perlu menggunakan findViewById, yang lebih aman dan mengurangi kemungkinan kesalahan. Dalam implementasi adapter RecyclerView, ItemGunungBinding digunakan untuk mengikat data ke tampilan item, memungkinkan akses langsung ke elemen UI yang ada dalam layout item_gunung.xml, seperti teks atau gambar, dengan cara yang lebih sederhana dan efisien. Pada baris [8] hingga [13] terdapat fungsi yang digunakan untuk mendefinisikan kelas GunungAdapter, yang merupakan adapter untuk RecyclerView di aplikasi Android. Kelas ini mengelola dan menampilkan data dalam bentuk daftar yang terdiri dari objek Gunung. Di dalam konstruktor, terdapat tiga parameter: listGunung, yang merupakan ArrayList<Gunung> yang berisi data gunung yang akan ditampilkan, onLinkClick, yang adalah fungsi untuk menangani aksi klik pada link yang menerima parameter bertipe String, dan onDetailClick, yang adalah fungsi untuk menangani klik pada elemen yang menampilkan detail gunung, dengan parameter yang mencakup ID dan tiga string lainnya (mungkin nama, lokasi, dan deskripsi gunung). Kelas ini mewarisi RecyclerView.Adapter dan menggunakan ListViewHolder sebagai ViewHolder untuk mengikat tampilan item.

Pada baris [15] hingga [30] fungsi ListViewHolder yang merupakan bagian dari adapter RecyclerView dan bertugas untuk mengikat data gunung ke tampilan item. Kelas ini menerima objek ItemGunungBinding, yang memungkinkan akses ke elemen-elemen UI dalam layout item secara langsung melalui ViewBinding. Di dalam fungsi bind(), data dari objek Gunung diikat ke elemen UI seperti nama gunung, lokasi, deskripsi, dan gambar. binding.tvGunungName.text diatur dengan nama gunung, binding.tvGunungLokasi.text diatur dengan lokasi, binding.tvGunungDeskripsi.text dengan deskripsi, dan binding.imgGunung.setImageResource digunakan untuk menampilkan gambar gunung berdasarkan resource ID. Selain itu, dua tombol diatur untuk menangani klik. Tombol binding.btnExit memiliki listener yang memanggil fungsi onLinkClick

dengan parameter link gunung ketika diklik, sementara tombol binding.btnDetail memiliki listener yang memanggil fungsi onDetailClick dengan parameter berupa informasi gunung seperti ID gambar, nama, lokasi, dan deskripsi ketika diklik. Pada baris [35] hingga [48] terdapat fungsi RecyclerView.Adapter yang bertugas untuk mengelola data dan menampilkan item di dalam RecyclerView. Fungsi onCreateViewHolder bertanggung jawab untuk membuat tampilan item dengan meng-inflate layout menggunakan ItemGunungBinding.inflate(), yang memungkinkan akses mudah ke elemen UI menggunakan ViewBinding. Fungsi ini kemudian mengembalikan objek ViewHolder, yang akan mengikat tampilan ke data yang sesuai. Fungsi getItemCount mengembalikan jumlah total item yang ada dalam daftar listGunung, yang memberi tahu RecyclerView berapa banyak item yang perlu ditampilkan. Sementara itu, fungsi onBindViewHolder dipanggil untuk mengikat data ke tampilan. Di dalam fungsi ini, data dari objek Gunung pada posisi tertentu diambil dan diteruskan ke metode bind() dalam ViewHolder, yang akan memperbarui elemen-elemen UI dengan informasi yang relevan. Selain itu, fungsi onLinkClick dan onDetailClick diteruskan untuk menangani interaksi pengguna seperti klik pada tombol.

30. BuilGradle.kts

Kode kelas ini secara keseluruhan berguna untuk engatur konfigurasi dan dependensi aplikasi. Pada baris [1] hingga [4] terdapat fungsi plugins mendeklarasikan plugin yang digunakan, yaitu plugin Android aplikasi, Kotlin Android, dan kotlin-parcelize untuk mempermudah pengiriman objek antar-komponen dengan Parcelable. Pada baris [7] hingga [19] terdapat fungsi android, ditentukan berbagai konfigurasi seperti namespace, versi compileSdk, serta pengaturan defaultConfig yang mencakup applicationId, minimum dan target SDK, versi aplikasi, serta runner untuk pengujian instrumentasi. Pada baris [22] hingga [28] terdapat fungsi buildTypes berisi konfigurasi untuk mode rilis, termasuk pengaturan ProGuard untuk optimisasi dan obfuscation kode.

Pada baris [33] hingga [39] terdapat fungsi compileOptions dan kotlinOptions menentukan bahwa proyek ini menggunakan Java 17 dan Kotlin dengan target JVM 17, menyesuaikan dengan fitur-fitur modern. Pada baris [42] dan [43] terdapat fungsi

buildFeatures yang diatur untuk mengaktifkan viewBinding, yang memudahkan akses elemen layout dalam kode Kotlin. Terakhir, pada baris [47] hingga [64] terdapat fungsi dependencies yang mencantumkan pustaka-pustaka penting yang digunakan dalam proyek, seperti core-ktx, appcompat, material, constraintlayout, recyclerview, cardview, dan lifecycle components. Selain itu, terdapat dependensi untuk unit testing (junit) dan pengujian instrumentasi (espresso dan androidx.test).

PEMBAHASAN DEBUGGING

- **Gambar 2. Debugger**

Debugger adalah sebuah alat bantu yang digunakan dalam proses pengembangan perangkat lunak untuk menemukan dan memperbaiki bug atau kesalahan dalam kode program. Dengan debugger, pengembang dapat menjalankan program secara bertahap (step-by-step), memantau nilai variabel, melihat alur eksekusi kode, dan menghentikan program pada titik tertentu (breakpoint) untuk menganalisis kondisi sistem pada saat itu. Fungsi utama debugger adalah untuk membantu pengembang memahami perilaku program secara mendetail dan mendeteksi letak kesalahan logika atau runtime error yang tidak terlihat hanya dari hasil eksekusi biasa.

Pada Gambar 2. Debugger, debugger sedang digunakan di dalam lingkungan pengembangan Android Studio. Terlihat bahwa debugger sedang berhenti (breakpoint) pada baris binding.tvGunungName.text = gunung.name, yang berarti eksekusi program berhenti sementara di baris tersebut untuk memungkinkan pengembang memeriksa nilai-nilai variabel. Di bagian bawah (panel Debug), dapat dilihat nilai objek gunung, yang merupakan instance dari kelas Gunung dengan informasi atribut seperti image, name, lokasi, dan deskripsi. Ini menunjukkan bahwa debugger sedang membantu pengembang memverifikasi apakah data yang dimasukkan ke dalam view (melalui binding) sesuai dengan nilai yang diharapkan. Dengan bantuan debugger ini, pengembang bisa memastikan bahwa proses binding data ke UI berjalan dengan benar, serta bisa menelusuri lebih lanjut jika terjadi kesalahan atau data tidak tampil sesuai harapan.

- **Gambar 3. Step Over (F8)**

Secara umum, "Step Over" adalah salah satu fitur dalam debugger yang digunakan untuk mengeksekusi satu baris kode saat ini dan kemudian berpindah ke baris berikutnya tanpa masuk ke dalam fungsi/metode yang dipanggil pada baris tersebut. Fitur ini sangat berguna ketika pengembang hanya ingin melihat hasil dari eksekusi baris itu secara langsung, tanpa perlu masuk dan melihat isi dari metode atau fungsi lain yang dipanggil. Hal ini mempercepat proses debugging jika pengembang sudah yakin bahwa fungsi yang dipanggil tersebut tidak mengandung bug atau tidak perlu dianalisis lebih dalam.

Pada Gambar 3. Step Over (F8) yang ditampilkan, debugger sedang berhenti pada baris `binding.tvGunungDeskripsi.text = gunung.deskripsi`, dan tombol "Step Over" (ikon panah ke bawah dengan garis horizontal di bawahnya) tersedia untuk digunakan. Jika pengembang mengklik tombol "Step Over", maka baris tersebut akan dieksekusi yang artinya teks deskripsi dari objek `gunung` akan di-set ke elemen `tvGunungDeskripsi` dan kemudian debugger akan melanjutkan ke baris berikutnya, yaitu `binding.imgGunung.setImageResource(gunung.image)`. Dalam konteks ini, "Step Over" digunakan untuk melihat secara bertahap bagaimana nilai dari properti objek `gunung` dipetakan ke tampilan UI tanpa masuk ke dalam fungsi setter atau ke dalam proses internal fungsi `setText()` yang sudah merupakan bagian dari Android SDK.

- **Gambar 4. Step Into (F7)**

Secara umum, "Step Into" adalah fitur dalam debugger yang digunakan untuk masuk ke dalam kode dari sebuah fungsi atau metode yang sedang dipanggil pada baris saat ini. Dengan kata lain, jika baris tersebut memanggil sebuah fungsi atau metode, maka menggunakan "Step Into" akan membawa debugger masuk ke dalam definisi fungsi tersebut, memungkinkan pengembang untuk melacak proses internalnya secara rinci. Ini sangat berguna saat pengembang ingin mengetahui secara persis bagaimana suatu fungsi bekerja atau ketika mencurigai bahwa bug terjadi di dalam fungsi itu.

Pada Gambar 4. Step Into (F7) yang ditampilkan, debugger sedang berhenti pada baris `binding.tvGunungLokasi.text = gunung.lokasi`. Jika pengembang menggunakan tombol "Step Into" pada titik ini (ikon panah ke bawah dengan garis melengkung ke dalam), debugger akan mencoba masuk ke dalam metode `setText()` milik objek `TextView` (dalam hal ini `tvGunungLokasi`). Artinya, debugger akan membawa kita ke dalam implementasi dari metode `setText()` milik Android SDK, jika kode sumbernya tersedia, atau ke dalam referensi internalnya. Namun, karena `setText()` merupakan bagian dari pustaka Android yang sudah stabil dan umum digunakan, biasanya pengembang hanya perlu menggunakan "Step Into" jika ingin menyelidiki lebih jauh fungsi-fungsi buatan sendiri, bukan dari framework. Dalam konteks gambar ini, penggunaan "Step Into" lebih cocok jika ingin memastikan bagaimana proses penyalinan data `gunung.lokasi` ke dalam tampilan dilakukan secara mendalam.

- **Gambar 5. Step Out (Shift + F8)**

Secara umum, "Step Out" adalah fitur dalam debugger yang digunakan untuk keluar dari fungsi atau metode yang sedang dijalankan saat ini, lalu melanjutkan eksekusi program hingga kembali ke pemanggil fungsi tersebut. Fitur ini sangat berguna jika pengembang sedang berada di dalam fungsi dan ingin segera kembali ke bagian kode yang memanggil fungsi itu, tanpa harus menelusuri setiap baris di dalam fungsi tersebut. "Step Out" sangat efisien saat pengembang menyadari bahwa tidak ada masalah di dalam fungsi saat ini dan ingin kembali ke level yang lebih tinggi dalam alur eksekusi.

Pada Gambar 5. Step Out (Shift + F8) yang ditampilkan, debugger sedang berada di dalam metode `onBindViewHolder` pada baris `holder.bind(getItem(position))`. Jika pengembang menggunakan tombol "Step Out" (ikon panah ke atas dari kotak), maka debugger akan mengeksekusi sisa baris dalam metode `onBindViewHolder()` ini secara langsung hingga selesai, dan kemudian melompat kembali ke kode yang memanggil `onBindViewHolder()`, biasanya dari dalam sistem RecyclerView Android. Dalam konteks ini, "Step Out"

akan mempercepat proses debugging karena pengembang mungkin sudah cukup melihat bagaimana data di-*bind* ke `viewHolder` dan ingin melanjutkan ke bagian alur logika yang lebih tinggi atau ke proses daur ulang tampilan berikutnya.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Jawab:

Dalam arsitektur aplikasi Android, Application class adalah salah satu komponen penting yang sering kali tidak disadari perannya oleh banyak developer, padahal fungsinya cukup vital. Secara sederhana, Application adalah kelas dasar yang digunakan untuk mempertahankan global state dari seluruh aplikasi. Android secara otomatis membuat instance dari kelas ini saat proses aplikasi pertama kali dimulai, sebelum aktivitas (Activity), service, atau komponen lainnya dipanggil. Fungsi utama dari Application class adalah sebagai tempat terbaik untuk menginisialisasi hal-hal yang bersifat global, seperti dependency injection (misalnya Hilt atau Dagger), konfigurasi library pihak ketiga (seperti Retrofit, Glide, Firebase, dsb), setup logging, atau menyimpan data yang ingin diakses lintas komponen dan aktivitas. Dengan kata lain, Application cocok digunakan untuk menaruh kode yang hanya ingin dijalankan sekali saat aplikasi pertama kali dibuka.

Biasanya, developer akan membuat subclass dari Application, lalu menambahkan kode inisialisasi di dalam onCreate(). Misalnya:

```
class MyApp : Application() {  
    override fun onCreate() {  
        super.onCreate()  
        // Inisialisasi global di sini  
        FirebaseApp.initializeApp(this)  
    }  
}
```

Setelah membuat kelas ini, kita harus mendeklarasikannya di file AndroidManifest.xml agar dikenali oleh sistem:

```
<application  
    android:name=".MyApp"  
    ... >
```

Jadi, Application class sangat berguna sebagai titik awal dalam siklus hidup

aplikasi Android, khususnya ketika kita ingin menyatukan pengaturan global agar lebih tertata dan efisien. Tapi perlu diingat, karena kelas ini bertahan sepanjang umur aplikasi, harus hati-hati dalam mengelola memori agar tidak menimbulkan memory leak.

MODUL 5 : Connect to the Internet

SOAL 1

Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

- a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
- b. Gunakan KotlinX Serialization sebagai library JSON.
- c. Gunakan library seperti Coil atau Glide untuk image loading.
- d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: <https://developer.themoviedb.org/docs/getting-started>
- e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
- f. Gunakan caching strategy pada Room..
- g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

A. Source Code

MODUL5/app/src/main/AndroidManifest.xml

1. AndroidManifest.xml

Tabel 36. Source Code AndroidManifest.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<manifest
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	package="com.example.modul5">
5	
6	<uses-permission
7	android:name="android.permission.ACCESS_NETWORK_STATE" />
8	<uses-permission android:name="android.permission.INTERNET"
9	/>
10	

```

11 <application
12     android:allowBackup="true"
13
14     android:dataExtractionRules="@xml/data_extraction_rules"
15         android:fullBackupContent="@xml/backup_rules"
16         android:icon="@mipmap/ic_launcher"
17         android:label="@string/app_name"
18         android:roundIcon="@mipmap/ic_launcher_round"
19         android:supportsRtl="true"
20         android:theme="@style/Theme.Modul5">
21
22     <activity
23         android:name=".MainActivity"
24         android:exported="true">
25         <intent-filter>
26             <action
27             android:name="android.intent.action.MAIN" />
28             <category
29             android:name="android.intent.category.LAUNCHER" />
30         </intent-filter>
31     </activity>
32
33 </application>
34
35 </manifest>

```

com/example/modul5/data/local/entity/dao

2. GunungDao

Tabel 37. Source Code GunungDao

1	package com.example.modul5.data.local.dao
2	
3	import androidx.room.*
4	import com.example.modul5.data.local.entity.GunungEntity
5	import kotlinx.coroutines.flow.Flow

```
6
7     @Dao
8     interface GunungDao {
9
10        @Query("SELECT * FROM gunung")
11        fun getAllGunung(): Flow<List<GunungEntity>>
12
13        @Query("SELECT * FROM gunung")
14        suspend fun getAllGunungOnce(): List<GunungEntity>
15
16        @Query("SELECT * FROM gunung WHERE isFavorite = 1")
17        suspend fun getAllFavoriteGunung(): List<GunungEntity>
18
19        @Query("SELECT * FROM gunung WHERE isFavorite = 1")
20        fun getAllFavoriteGunungFlow(): Flow<List<GunungEntity>>
21
22        @Insert(onConflict = OnConflictStrategy.REPLACE)
23        suspend fun insertGunungList(gunungList:
24            List<GunungEntity>)
25
26        @Insert(onConflict = OnConflictStrategy.REPLACE)
27        suspend fun insertGunung(gunung: GunungEntity)
28
29        @Query("DELETE FROM gunung")
30        suspend fun clearGunung()
31
32        @Delete
33        suspend fun delete(gunung: GunungEntity)
34
35        @Update
36        suspend fun update(gunung: GunungEntity)
37
38        @Query("UPDATE gunung SET isFavorite = :isFavorite WHERE
39        name = :name")
```

```

40     suspend fun updateFavoriteStatus(name: String, isFavorite:
41         Boolean)
42
43 }
```

com/example/modul5/data/local/entity/database

3. AppDatabase

Tabel 38. Source Code AppDatabase

```

1 package com.example.modul5.data.local.database
2
3 import android.content.Context
4 import androidx.room.Database
5 import androidx.room.Room
6 import androidx.room.RoomDatabase
7 import com.example.modul5.data.local.dao.GunungDao
8 import com.example.modul5.data.local.entity.GunungEntity
9
10 @Database(entities = [GunungEntity::class], version = 1,
11 exportSchema = false)
12 abstract class AppDatabase : RoomDatabase() {
13
14     abstract fun gunungDao(): GunungDao
15
16     companion object {
17         @Volatile
18         private var INSTANCE: AppDatabase? = null
19
20         fun getDatabase(context: Context): AppDatabase {
21             return INSTANCE ?: synchronized(this) {
22                 val instance = Room.databaseBuilder(
23                     context.applicationContext,
24                     AppDatabase::class.java,
25                     "gunung_database"
26                 ).build()
27             }
28         }
29     }
30 }
```

```

27             INSTANCE = instance
28
29         }
30     }
31 }
32 }
```

com/example/modul5/data/local/entity/entity

4. GunungEntity.kt

Tabel 39. Source Code GunungEntity.kt

```

1 package com.example.modul5.data.local.entity
2
3 import androidx.room.Entity
4 import androidx.room.PrimaryKey
5
6 @Entity(tableName = "gunung")
7 data class GunungEntity(
8     @PrimaryKey val name: String,
9     val lokasi: String,
10    val deskripsi: String,
11    val link: String,
12    val image: String,
13    val isFavorite: Boolean
14 )
```

com/example/modul5/data/mapper

5. GunungMapper.kt

Tabel 40. Source Code GunungMapper.kt

```

1 package com.example.modul5.data.mapper
2
3 import com.example.modul5.data.model.Gunung
4 import com.example.modul5.data.local.entity.GunungEntity
5
6 fun Gunung.toEntity(): GunungEntity {
```

```

7     return GunungEntity(
8         name = name,
9         lokasi = lokasi,
10        deskripsi = deskripsi,
11        link = link,
12        image = image,
13        isFavorite = isFavorite
14    )
15 }
16
17 fun GunungEntity.toModel(): Gunung {
18     return Gunung(
19         name = name,
20         lokasi = lokasi,
21         deskripsi = deskripsi,
22         link = link,
23         image = image,
24         isFavorite = isFavorite
25     )
26 }
```

com/example/modul5/data/model

6. Gunung

Tabel 41. Source Code Gunung

1	package com.example.modul5.data.model
2	
3	import kotlinx.serialization.SerializedName
4	import kotlinx.serialization.Serializable
5	import java.io.Serializable as JavaSerializable
6	
7	@Serializable
8	data class Gunung(
9	@SerializedName("name")
10	val name: String,

```

11
12     @SerializedName("lokasi")
13     val lokasi: String,
14
15     @SerializedName("deskripsi")
16     val deskripsi: String,
17
18     @SerializedName("link")
19     val link: String,
20
21     @SerializedName("image_url")
22     val image: String,
23
24     val isFavorite: Boolean = false
25 ) : JavaSerializable

```

com/example/modul5/data/remote

7. ApiClient

Tabel 42. Source Code ApiClient

```

1 package com.example.modul5.data.remote
2
3 import kotlinx.serialization.json.Json
4 import okhttp3.MediaType.Companion.toMediaType
5 import okhttp3.OkHttpClient
6 import okhttp3.logging.HttpLoggingInterceptor
7 import retrofit2.Retrofit
8 import
9 com.jakewharton.retrofit2.converter.kotlinx.serialization.asCo
10 nverterFactory
11
12 import java.util.concurrent.TimeUnit
13
14 object ApiClient {
15     private const val BASE_URL =

```

```

16 "https://modul5.free.beeceptor.com/api/path/"
17
18     private val json = Json {
19         ignoreUnknownKeys = true
20     }
21
22     private val loggingInterceptor =
23 HttpLoggingInterceptor().apply {
24         level = HttpLoggingInterceptor.Level.BODY
25     }
26
27     private val okHttpClient = OkHttpClient.Builder()
28         .addInterceptor(loggingInterceptor)
29         .connectTimeout(30, TimeUnit.SECONDS)
30         .readTimeout(30, TimeUnit.SECONDS)
31         .build()
32
33     val retrofit: Retrofit = Retrofit.Builder()
34         .baseUrl(BASE_URL)
35         .client(okHttpClient)
36
37     .addConverterFactory(json.asConverterFactory("application/json
38 ".toMediaType()))
39         .build()
40
41     val gunung ApiService: Gunung ApiService =
42 retrofit.create(Gunung ApiService::class.java)
43 }
```

8. Gunung ApiService

Tabel 43. Source Code Gunung ApiService

1	package com.example.modul5.data.remote
2	
3	import com.example.modul5.data.model.Gunung

```

4 import retrofit2.http.GET
5
6 interface Gunung ApiService {
7     @GET("gunung")
8     suspend fun getGunungList(): List<Gunung>
9 }

```

[com/example/modul5/data/repository](#)

9. GunungRepository

Tabel 44. Source Code GunungRepository

```

1 package com.example.modul5.data.repository
2
3 import android.content.Context
4 import com.example.modul5.data.local.database.AppDatabase
5 import com.example.modul5.data.mapper.toEntity
6 import com.example.modul5.data.mapper.toModel
7 import com.example.modul5.data.model.Gunung
8 import com.example.modul5.data.remote.ApiClient
9 import kotlinx.coroutines.Dispatchers
10 import kotlinx.coroutines.flow.Flow
11 import kotlinx.coroutines.flow.map
12 import kotlinx.coroutines.withContext
13
14 class GunungRepository(context: Context) {
15
16     private val apiService = ApiClient.gunung ApiService
17     private val gunungDao =
18 AppDatabase.getDatabase(context).gunung Dao()
19
20     fun getFavoriteGunungListFlow(): Flow<List<Gunung>> {
21         return gunungDao.getAllFavoriteGunungFlow()
22             .map { it.map { entity -> entity.toModel() } }
23     }
24

```

```
25     suspend fun fetchGunungList(): List<Gunung> =
26     withContext(Dispatchers.IO) {
27         try {
28             val remoteList = apiService.getGunungList()
29             val favoriteNames =
30             gunungDao.getAllFavoriteGunung().map { it.name }
31
32             val mergedList = remoteList.map { gunung ->
33                 if (gunung.name in favoriteNames)
34                     gunung.copy(isFavorite = true)
35                 else gunung
36             }
37
38             gunungDao.clearGunung()
39             gunungDao.insertGunungList(mergedList.map {
40                 it.toEntity() })
41
42             return@withContext mergedList
43         } catch (e: Exception) {
44             return@withContext gunungDao.getAllGunungOnce().map
45             { it.toModel() }
46         }
47     }
48
49     suspend fun updateGunung(gunung: Gunung) =
50     withContext(Dispatchers.IO) {
51         gunungDao.update(gunung.toEntity())
52     }
53
54     suspend fun insertFavoriteGunung(gunung: Gunung) =
55     withContext(Dispatchers.IO) {
56         gunungDao.updateFavoriteStatus(gunung.name, true)
57     }
58 }
```

```

59     suspend fun deleteFavoriteGunung(gunung: Gunung) =
60     withContext(Dispatchers.IO) {
61         gunungDao.updateFavoriteStatus(gunung.name, false)
62     }
63 }
```

10. RepositoryInstance

Tabel 45. Source Code RepositoryInstance

```

1 package com.example.modul5.data.repository
2
3 import android.content.Context
4
5 object RepositoryInstance {
6     private var repository: GunungRepository? = null
7
8     fun provideRepository(context: Context): GunungRepository {
9         return repository ?: synchronized(this) {
10             val instance =
11                 GunungRepository(context.applicationContext)
12             repository = instance
13             instance
14         }
15     }
16 }
```

com/example/modul5/ui

11. FavoriteGunungFragment.kt

Tabel 46. Source Code FavoriteGunungFragment.kt

```

1 package com.example.modul5.ui
2
3 import android.content.Intent
4 import android.net.Uri
5 import androidx.fragment.app.viewModels
6 import android.os.Bundle
```

```
7 import android.view.LayoutInflater
8 import android.view.View
9 import android.view.ViewGroup
10 import androidx.fragment.app.Fragment
11 import androidx.lifecycle.lifecycleScope
12 import androidx.recyclerview.widget.LinearLayoutManager
13 import com.example.modul5.R
14 import
15 com.example.modul5.databinding.FragmentFavoritGunungBinding
16 import com.example.modul5.viewmodel.GunungViewModel
17 import com.example.modul5.viewmodel.GunungViewModelFactory
18 import com.example.modul5.data.repository.RepositoryInstance
19 import kotlinx.coroutines.launch
20
21 class FavoritGunungFragment : Fragment() {
22
23     private var _binding: FragmentFavoritGunungBinding? =
24 null
25     private val binding get() = _binding!!
26
27     private lateinit var adapter: GunungAdapter
28
29     private val viewModel: GunungViewModel by viewModels {
30
31         GunungViewModelFactory(RepositoryInstance.provideRepository(
32             requireContext()))
33     }
34
35     override fun onCreateView(
36         inflater: LayoutInflater, container: ViewGroup?,
37         savedInstanceState: Bundle?
38     ): View {
39         _binding =
40         FragmentFavoritGunungBinding.inflate(inflater, container,
```

```
41    false)
42
43        return binding.root
44
45    override fun onViewCreated(view: View,
46 savedInstanceState: Bundle?) {
47
48        super.onViewCreated(view, savedInstanceState)
49
50        adapter = GunungAdapter(
51            onLinkClick = { gunung ->
52                val intent = Intent(Intent.ACTION_VIEW,
53                Uri.parse(gunung.link))
54                startActivity(intent)
55            },
56            onDetailClick = { gunung ->
57                val detail = GunungDetailFragment().apply {
58                    arguments = Bundle().apply {
59                        putString("EXTRA_NAME", gunung.name)
60                        putString("EXTRA_LOKASI",
61                            gunung.lokasi)
62                        putString("EXTRA_DESKRIPSI",
63                            gunung.deskripsi)
64                        putString("EXTRA_PHOTO",
65                            gunung.image)
66                    }
67                }
68                parentFragmentManager.beginTransaction()
69                .replace(R.id.fragmentContainerGunung,
70                detail)
71                .addToBackStack(null)
72                .commit()
73
74        requireActivity().findViewById<View>(R.id.fragmentContainerG
```

```

75     unung).visibility = View.VISIBLE
76         }
77         ,
78         onFavoriteClick = { gunung ->
79             val updatedGunung = gunung.copy(isFavorite =
80             !gunung.isFavorite)
81             viewModel.updateGunung(updatedGunung)
82         }
83     )
84
85     binding.recyclerViewFavorit.layoutManager =
86     LinearLayoutManager(requireContext())
87     binding.recyclerViewFavorit.adapter = adapter
88
89     lifecycleScope.launch {
90         viewModel.favoriteList.collect { favorites ->
91             adapter.submitList(favorites)
92         }
93     }
94 }
95
96     override fun onDestroyView() {
97         super.onDestroyView()
98         _binding = null
99     }
100 }
```

12. GunungAdapter

Tabel 47. Source Code GunungAdapter

1	package com.example.modul5.ui
2	
3	import android.view.LayoutInflater
4	import android.view.ViewGroup
5	import androidx.recyclerview.widget.DiffUtil

```
6 import androidx.recyclerview.widget.ListAdapter
7 import androidx.recyclerview.widget.RecyclerView
8 import coil.load
9 import com.example.modul5.R
10 import com.example.modul5.data.model.Gunung
11 import com.example.modul5.databinding.ItemGunungBinding
12
13 class GunungAdapter(
14     private val onLinkClick: (Gunung) -> Unit,
15     private val onDetailClick: (Gunung) -> Unit,
16     private val onFavoriteClick: (Gunung) -> Unit
17 ) : ListAdapter<Gunung,
18 GunungAdapter.ListViewHolder>(DIFF_CALLBACK) {
19
20     inner class ListViewHolder(private val binding:
21 ItemGunungBinding) :
22 RecyclerView.ViewHolder(binding.root) {
23
24         fun bind(gunung: Gunung) {
25             binding.tvGunungName.text = gunung.name
26             binding.tvGunungLokasi.text = gunung.lokasi
27             binding.tvGunungDeskripsi.text = gunung.deskripsi
28             binding.imgGunung.load(gunung.image)
29
30             val iconRes = if (gunung.isFavorite)
31 R.drawable.ic_star_filled else R.drawable.ic_star_border
32             binding.btnFavorite.setImageResource(iconRes)
33
34             binding.btnLink.setOnClickListener {
35                 onLinkClick(gunung)
36             }
37             binding.btnDetail.setOnClickListener {
38                 onDetailClick(gunung)
39             }

```

```

40         binding.btnExit.setOnClickListener {
41             onFavoriteClick(gunung)
42         }
43     }
44 }
45
46     override fun onCreateViewHolder(parent: ViewGroup,
47 viewType: Int): ListViewHolder {
48         val binding =
49             ItemGunungBinding.inflate(LayoutInflater.from(parent.context),
50             parent, false)
51         return ListViewHolder(binding)
52     }
53
54     override fun onBindViewHolder(holder: ListViewHolder,
55 position: Int) {
56         holder.bind(getItem(position))
57     }
58
59     companion object {
60         private val DIFF_CALLBACK = object :
61             DiffUtil.ItemCallback<Gunung>() {
62             override fun areItemsTheSame(oldItem: Gunung,
63             newItem: Gunung) =
64                 oldItem.name == newItem.name
65
66             override fun areContentsTheSame(oldItem: Gunung,
67             newItem: Gunung) =
68                 oldItem == newItem
69         }
70     }
71 }

```

13. GunungDetailFragment

Tabel 48. Source Code GunungDetailFragment

```
1 package com.example.modul5.ui
2
3 import android.os.Bundle
4 import androidx.fragment.app.Fragment
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.view.ViewGroup
8 import coil.load
9 import com.example.modul5.R
10 import com.example.modul5.databinding.DetailFragmentBinding
11
12 class GunungDetailFragment : Fragment() {
13
14     private var _binding: DetailFragmentBinding? = null
15     private val binding get() = _binding!!
16
17     override fun onCreateView(
18         inflater: LayoutInflater, container: ViewGroup?,
19         savedInstanceState: Bundle?
20     ): View {
21         _binding = DetailFragmentBinding.inflate(inflater, container,
22         false)
23
24         val name = arguments?.getString("EXTRA_NAME") ?: "Tidak ada
25         nama"
26         val lokasi = arguments?.getString("EXTRA_LOKASI") ?: "Tidak
27         ada lokasi"
28         val deskripsi = arguments?.getString("EXTRA_DESKRIPSI") ?:
29         "Deskripsi belum tersedia"
30         val image = arguments?.getString("EXTRA_PHOTO")
31
32         binding.tvName.text = name
33         binding.tvLokasi.text = lokasi
34         binding.tvDeskripsi.text = deskripsi
35
36         image?.let {
37             binding.imgPoster.load(it) {
38                 crossfade(true)
39                 placeholder(android.R.drawable.ic_menu_gallery)
40                 error(android.R.drawable.ic_menu_report_image)
41             }
42         }
43
44         binding.btnExit.setOnClickListener {
45             parentFragmentManager.popBackStack()
46
47         activity?.findViewById<View>(R.id.fragmentContainerGunung)?.visibility
48         = View.GONE
49     }
50
51     return binding.root
52 }
```

```

53
54     override fun onDestroyView() {
55         super.onDestroyView()
56         _binding = null
57     }
58 }
```

14. GunungListFragment

Tabel 49. Source Code GunungListFragment

```

1 package com.example.modul5.ui
2
3 import android.content.Intent
4 import android.net.Uri
5 import android.os.Bundle
6 import android.view.LayoutInflater
7 import android.view.View
8 import android.view.ViewGroup
9 import android.widget.Toast
10 import androidx.fragment.app.Fragment
11 import androidx.lifecycle.ViewModelProvider
12 import androidx.lifecycle.lifecycleScope
13 import androidx.recyclerview.widget.LinearLayoutManager
14 import com.example.modul5.R
15 import com.example.modul5.data.repository.GunungRepository
16 import com.example.modul5.databinding.ListFragmentBinding
17 import com.example.modul5.viewmodel.GunungViewModel
18 import com.example.modul5.viewmodel.GunungViewModelFactory
19 import kotlinx.coroutines.flow.collect
20
21 class GunungListFragment : Fragment() {
22
23     private var _binding: ListFragmentBinding? = null
24     private val binding get() = _binding!!
25
26     private lateinit var viewModel: GunungViewModel
27     private lateinit var adapter: GunungAdapter
28 }
```

```
29     override fun onCreate(savedInstanceState: Bundle?) {
30         super.onCreate(savedInstanceState)
31         val repository = GunungRepository(requireContext())
32         val factory = GunungViewModelFactory(repository)
33         viewModel = ViewModelProvider(this,
34         factory) [GunungViewModel::class.java]
35     }
36
37     override fun onCreateView(
38         inflater: LayoutInflater,
39         container: ViewGroup?,
40         savedInstanceState: Bundle?
41     ): View {
42         _binding = ListFragmentBinding.inflate(inflater,
43         container, false)
44         return binding.root
45     }
46
47     override fun onViewCreated(view: View,
48         savedInstanceState: Bundle?) {
49         super.onViewCreated(view, savedInstanceState)
50
51         adapter = GunungAdapter(
52             onLinkClick = { gunung ->
53                 val intent = Intent(Intent.ACTION_VIEW,
54                 Uri.parse(gunung.link))
55                 startActivity(intent)
56             },
57             onDetailClick = { gunung ->
58                 viewModel.selectGunung(gunung)
59             },
60             onFavoriteClick = { gunung ->
61                 val updatedGunung = gunung.copy(isFavorite =
62                 !gunung.isFavorite)
```

```
63             viewModel.updateGunung(updatedGunung)
64         }
65     )
66
67     binding.rvGunung.layoutManager =
68 LinearLayoutManager(requireContext())
69     binding.rvGunung.adapter = adapter
70
71     lifecycleScope.launchWhenStarted {
72         viewModel.gunungList.collect { listGunung ->
73             adapter.submitList(listGunung)
74         }
75     }
76
77     lifecycleScope.launchWhenStarted {
78         viewModel.errorMessage.collect { error ->
79             error?.let {
80                 Toast.makeText(requireContext(), it,
81                 Toast.LENGTH_SHORT).show()
82             }
83         }
84     }
85
86     lifecycleScope.launchWhenStarted {
87         viewModel.selectedGunung.collect { selected ->
88             selected?.let {
89                 val detailFragment =
90                 GunungDetailFragment().apply {
91                     arguments = Bundle().apply {
92                         putString("EXTRA_NAME", it.name)
93                         putString("EXTRA_LOKASI",
94                         it.lokasi)
95                         putString("EXTRA_DESKRIPSI",
96                         it.deskripsi)

```

```

97                     putString("EXTRA_PHOTO",
98             it.image)
99         }
100     }
101
102
103 requireActivity().supportFragmentManager.beginTransaction()
104
105 .replace(R.id.fragmentContainerGunung, detailFragment)
106         .addToBackStack(null)
107         .commit()
108
109
110 requireActivity().findViewById<View>(R.id.fragmentContainerG
111 unung).visibility = View.VISIBLE
112         }
113     }
114 }
115
116
117 override fun onDestroyView() {
118     super.onDestroyView()
119     _binding = null
120 }
121 }
123
7

```

15. ViewPagerAdapter

Tabel 50. Source Code ViewPagerAdapter

1	package com.example.modul5.ui
2	
3	import androidx.fragment.app.Fragment
4	import androidx.fragment.app.FragmentActivity
5	import androidx.viewpager2.adapter.FragmentStateAdapter

```

6
7 class ViewPagerAdapter(activity: FragmentActivity) :
8 FragmentStateAdapter(activity) {
9
10    private val fragments = listOf(
11        GunungListFragment(),
12        FavoritGunungFragment()
13    )
14
15    private val fragmentTitles = listOf(
16        "List Gunung",
17        "Favorit"
18    )
19
20    override fun getItemCount(): Int = fragments.size
21
22    override fun createFragment(position: Int): Fragment =
23        fragments[position]
24
25    fun getPageTitle(position: Int): String =
26        fragmentTitles[position]
27 }
```

com/example/modul5/viewmodel

16. GunungViewModel

Tabel 51. Source Code GunungViewModel

1	package com.example.modul5.viewmodel
2	
3	import android.util.Log
4	import androidx.lifecycle.ViewModel
5	import androidx.lifecycle.viewModelScope
6	import com.example.modul5.data.model.Gunung
7	import com.example.modul5.data.repository.GunungRepository
8	import kotlinx.coroutines.flow.*

```
9 import kotlinx.coroutines.launch
10
11 class GunungViewModel(private val repository: GunungRepository)
12 : ViewModel() {
13
14     private val _gunungList =
15     MutableStateFlow<List<Gunung>>(emptyList())
16     val gunungList: StateFlow<List<Gunung>> =
17     _gunungList.asStateFlow()
18
19     val favoriteList: StateFlow<List<Gunung>> =
20     repository.getFavoriteGunungListFlow()
21         .stateIn(viewModelScope,
22 SharingStarted.WhileSubscribed(5000), emptyList())
23
24     private val _selectedGunung =
25     MutableStateFlow<Gunung?>(null)
26     val selectedGunung: StateFlow<Gunung?> =
27     _selectedGunung.asStateFlow()
28
29     private val _errorMessage = MutableStateFlow<String?>(null)
30     val errorMessage: StateFlow<String?> =
31     _errorMessage.asStateFlow()
32
33     init {
34         fetchGunungList()
35     }
36
37     private fun fetchGunungList() {
38         viewModelScope.launch {
39             try {
40                 val remoteList = repository.fetchGunungList()
41                 _gunungList.value = remoteList
42             } catch (e: Exception) {
```

```
43             _errorMessage.value = "Gagal memuat data:  
44             ${e.localizedMessage}"  
45         }  
46     }  
47 }  
48  
49     fun selectGunung(gunung: Gunung) {  
50         _selectedGunung.value = gunung  
51         Log.d("GunungViewModel", "Gunung dipilih:  
52             ${gunung.name}")  
53     }  
54  
55     fun updateGunung(updatedGunung: Gunung) {  
56         viewModelScope.launch {  
57             try {  
58                 if (updatedGunung.isFavorite) {  
59  
60                     repository.insertFavoriteGunung(updatedGunung)  
61                 } else {  
62  
63                     repository.deleteFavoriteGunung(updatedGunung)  
64                 }  
65  
66                     _gunungList.value = _gunungList.value.map {  
67                         if (it.name == updatedGunung.name)  
68                             updatedGunung else it  
69                     }  
70  
71                     Log.d("GunungViewModel", "Gunung diperbarui dan  
72                     disimpan: ${updatedGunung.name}, Favorite:  
73                     ${updatedGunung.isFavorite}")  
74                 } catch (e: Exception) {  
75                     _errorMessage.value = "Gagal menyimpan favorit:  
76                     ${e.localizedMessage}"
```

```

77         }
78     }
79 }
80
81     fun onLinkClicked(gunung: Gunung) {
82         Log.d("GunungViewModel", "Explicit intent diklik untuk:
83 ${gunung.name}")
84     }
85 }
```

17. GunungViewModelFactory

Tabel 52. Source Code GunungViewModelFactory

```

1 package com.example.modul5.viewmodel
2
3 import androidx.lifecycle.ViewModel
4 import androidx.lifecycle.ViewModelProvider
5 import com.example.modul5.data.repository.GunungRepository
6
7 class GunungViewModelFactory(private val repository:
8 GunungRepository) : ViewModelProvider.Factory {
9     override fun <T : ViewModel> create(modelClass: Class<T>):
10 T {
11     if
12 (modelClass.isAssignableFrom(GunungViewModel::class.java)) {
13         @Suppress ("UNCHECKED_CAST")
14         return GunungViewModel(repository) as T
15     }
16     throw IllegalArgumentException("Unknown ViewModel
17 class")
18 }
19 }
```

[com/example/modul5/MainActivity.kt](#)

18. MainActivity

Tabel 53. Source Code MainActivity

```
1 package com.example.modul5
2
3 import android.os.Bundle
4 import androidx.appcompat.app.AppCompatActivity
5 import com.example.modul5.databinding.ActivityMainBinding
6 import com.example.modul5.ui.ViewPagerAdapter
7 import com.google.android.material.tabs.TabLayoutMediator
8
9 class MainActivity : AppCompatActivity() {
10
11     private lateinit var binding: ActivityMainBinding
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15
16         binding = ActivityMainBinding.inflate(layoutInflater)
17         setContentView(binding.root)
18
19         val viewPagerAdapter = ViewPagerAdapter(this)
20         binding.viewPager.adapter = viewPagerAdapter
21
22         TabLayoutMediator(binding.tabLayout, binding.viewPager)
23     { tab, position ->
24         tab.text = viewPagerAdapter.getPageTitle(position)
25     }.attach()
26     }
27 }
```

MODUL5/app/src/main/res/layout

19. activity_main.xml

Tabel 54. Source Code activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
```

```
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <com.google.android.material.tabs.TabLayout
9         android:id="@+id/tabLayout"
10        android:layout_width="0dp"
11        android:layout_height="wrap_content"
12        app:layout_constraintTop_toTopOf="parent"
13        app:layout_constraintStart_toStartOf="parent"
14        app:layout_constraintEnd_toEndOf="parent"
15        app:tabIndicatorColor="@color/blue_500"
16        app:tabSelectedTextColor="@color/blue_500"
17        app:tabTextColor="@color/black" />
18
19     <androidx.viewpager2.widget.ViewPager2
20         android:id="@+id/viewPager"
21         android:layout_width="0dp"
22         android:layout_height="0dp"
23         app:layout_constraintTop_toBottomOf="@+id/tabLayout"
24         app:layout_constraintBottom_toBottomOf="parent"
25         app:layout_constraintStart_toStartOf="parent"
26         app:layout_constraintEnd_toEndOf="parent"/>
27
28     <FrameLayout
29         android:id="@+id/fragmentContainerGunung"
30         android:layout_width="0dp"
31         android:layout_height="0dp"
32         android:visibility="gone"
33         app:layout_constraintTop_toBottomOf="@+id/tabLayout"
34         app:layout_constraintBottom_toBottomOf="parent"
35         app:layout_constraintStart_toStartOf="parent"
36         app:layout_constraintEnd_toEndOf="parent"/>
37
```

38

</androidx.constraintlayout.widget.ConstraintLayout>

20. detail_fragment.xml

Tabel 55. Source Code detail_fragment.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ScrollView
3     xmlns:android="http://schemas.android.com/apk/res/android"
4         xmlns:tools="http://schemas.android.com/tools"
5             android:layout_width="match_parent"
6             android:layout_height="match_parent"
7             android:background="#B6D6F1"
8             android:padding="16dp"
9             tools:context="com.example.modul5.ui.GunungDetailFragment">
10
11     <LinearLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:orientation="vertical">
15
16         <ImageView
17             android:id="@+id/imgPoster"
18             android:layout_width="match_parent"
19             android:layout_height="300dp"
20             android:contentDescription="Foto Gunung"
21             android:scaleType="centerCrop" />
22
23         <TextView
24             android:id="@+id/tvName"
25             android:layout_width="match_parent"
26             android:layout_height="wrap_content"
27             android:gravity="center"
28             android:textSize="35sp"
29             android:textStyle="bold"
30             android:textColor="@android:color/black"
```

```
31         android:layout_marginTop="12dp"
32         tools:text="Gunung Semeru" />
33
34     <TextView
35         android:id="@+id/tvLokasi"
36         android:layout_width="match_parent"
37         android:layout_height="wrap_content"
38         android:gravity="center"
39         android:textSize="22sp"
40         android:layout_marginTop="8dp"
41         tools:text="Lokasi: Jawa Timur" />
42
43     <TextView
44         android:id="@+id/tvDeskripsi"
45         android:layout_width="match_parent"
46         android:layout_height="wrap_content"
47         android:textSize="18sp"
48         android:layout_marginTop="8dp"
49         android:justificationMode="inter_word"
50         tools:text="Gunung tertinggi di Pulau Jawa yang
51 terkenal dengan jalur pendakian yang menantang dan keindahan
52 pemandangan alam." />
53
54     <Button
55         android:id="@+id/btnBack"
56         android:layout_width="wrap_content"
57         android:layout_height="wrap_content"
58         android:text="Kembali"
59         android:layout_marginTop="16dp"
60         android:layout_gravity="center" />
61     </LinearLayout>
62 </ScrollView>
63
64
```

21. fragment_favorit_gunung.xml

Tabel 56. Source Code fragment_favorit_gunung.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent">
7
8     <androidx.recyclerview.widget.RecyclerView
9         android:id="@+id/recyclerViewFavorit"
10        android:layout_width="0dp"
11        android:layout_height="0dp"
12        app:layout_constraintTop_toTopOf="parent"
13        app:layout_constraintBottom_toBottomOf="parent"
14        app:layout_constraintStart_toStartOf="parent"
15        app:layout_constraintEnd_toEndOf="parent"/>
16 </androidx.constraintlayout.widget.ConstraintLayout>
```

22. item_gunung.xml

Tabel 57. Source Code item_gunung.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.cardview.widget.CardView
3
4     xmlns:android="http://schemas.android.com/apk/res/android"
5         xmlns:app="http://schemas.android.com/apk/res-auto"
6         xmlns:tools="http://schemas.android.com/tools"
7         android:layout_width="match_parent"
8         android:layout_height="wrap_content"
9         android:layout_margin="8dp"
10        app:cardElevation="4dp"
11        app:cardCornerRadius="8dp">
12
13     <androidx.constraintlayout.widget.ConstraintLayout
```

```
14         android:layout_width="match_parent"
15         android:layout_height="wrap_content"
16         android:background="@color/cream"
17         android:padding="16dp">
18
19     <ImageView
20         android:id="@+id/imgGunung"
21         android:layout_width="120dp"
22         android:layout_height="160dp"
23         android:scaleType="centerCrop"
24         app:layout_constraintTop_toTopOf="parent"
25         app:layout_constraintStart_toStartOf="parent"
26         app:layout_constraintBottom_toBottomOf="parent"
27     />
28
29     <TextView
30         android:id="@+id/tvGunungName"
31         android:layout_width="0dp"
32         android:layout_height="wrap_content"
33         android:layout_marginStart="12dp"
34         android:textColor="@android:color/black"
35         android:textSize="24sp"
36         android:textStyle="bold"
37         app:layout_constraintTop_toTopOf="@+id/imgGunung"
38
39         app:layout_constraintStart_toEndOf="@+id/imgGunung"
40             app:layout_constraintEnd_toEndOf="parent"
41             tools:text="Gunung Semeru" />
42
43     <TextView
44         android:id="@+id/tvGunungLokasi"
45         android:layout_width="0dp"
46         android:layout_height="wrap_content"
47         android:layout_marginTop="4dp"
```

```
48         android:textColor="#666666"
49         android:textSize="14sp"
50         android:textStyle="bold"
51
52     app:layout_constraintTop_toBottomOf="@+id/tvGunungName"
53
54     app:layout_constraintStart_toStartOf="@+id/tvGunungName"
55
56     app:layout_constraintEnd_toEndOf="@+id/tvGunungName"
57         tools:text="Lokasi: Jawa Timur" />
58
59     <TextView
60         android:id="@+id/tvGunungDeskripsi"
61         android:layout_width="0dp"
62         android:layout_height="wrap_content"
63         android:layout_marginTop="4dp"
64         android:ellipsize="end"
65         android:maxLines="2"
66         android:textColor="@android:color/black"
67         android:textSize="14sp"
68
69     app:layout_constraintTop_toBottomOf="@+id/tvGunungLokasi"
70
71     app:layout_constraintStart_toStartOf="@+id/tvGunungLokasi"
72
73     app:layout_constraintEnd_toEndOf="@+id/tvGunungLokasi"
74         tools:text="Gunung tertinggi di Jawa Timur
75 dengan pemandangan yang sangat indah dan memiliki trek yang
76 menantang..." />
77
78     <androidx.constraintlayout.widget.Barrier
79         android:id="@+id/barrierStartText"
80         app:barrierDirection="start"
81
```

```
82    app:constraint_referenced_ids="tvGunungName, tvGunungLokasi, t
83    vGunungDeskripsi"
84            android:layout_width="wrap_content"
85            android:layout_height="wrap_content" />
86
87        <Button
88            android:id="@+id	btnLink"
89            android:layout_width="wrap_content"
90            android:layout_height="48dp"
91            android:minWidth="70dp"
92            android:layout_marginTop="12dp"
93            android:layout_marginEnd="4dp"
94            android:backgroundTint="@color/teal_700"
95            android:text="@string/btn_link_text"
96            android:textColor="@android:color/white"
97            android:textSize="12sp"
98            android:textStyle="bold"
99            android:includeFontPadding="false"
100
101        app:layout_constraintTop_toBottomOf="@+id/tvGunungDeskripsi"
102
103        app:layout_constraintStart_toStartOf="@+id/barrierStartText"
104
105        app:layout_constraintEnd_toStartOf="@+id/btnDetail"
106
107        app:layout_constraintBottom_toBottomOf="@+id/btnDetail" />
108
109        <Button
110            android:id="@+id	btnDetail"
111            android:layout_width="wrap_content"
112            android:layout_height="48dp"
113            android:minWidth="100dp"
114            android:layout_marginEnd="4dp"
115            android:translationY="-12dp"
```

```
116         android:backgroundTint="@color/blue"
117         android:text="@string/btn_detail_text"
118         android:textColor="@android:color/white"
119         android:textSize="12sp"
120         android:textStyle="bold"
121         android:includeFontPadding="false"
122         app:layout_constraintTop_toTopOf="@+id/btnLink"
123
124     app:layout_constraintBottom_toBottomOf="@+id/btnLink"
125         app:layout_constraintStart_toEndOf="@+id/btnLink"
126
127     app:layout_constraintEnd_toStartOf="@+id/btnFavorite" />
128
129
130     <ImageButton
131         android:id="@+id/btnFavorite"
132         android:layout_width="48dp"
133         android:layout_height="48dp"
134         android:layout_marginEnd="0dp"
135         android:backgroundTint="@color/pink"
136
137     android:background="?attr/selectableItemBackgroundBorderless"
138 "
139
140     android:contentDescription="@string/btn_favorite_text"
141         android:padding="8dp"
142         android:scaleType="centerInside"
143         android:src="@drawable/ic_star_border"
144         app:layout_constraintTop_toTopOf="@+id/btnLink"
145
146     app:layout_constraintBottom_toBottomOf="@+id/btnLink"
147         app:layout_constraintEnd_toEndOf="parent" />
148
149
```

150	</androidx.constraintlayout.widget.ConstraintLayout>
1	</androidx.cardview.widget.CardView>

23. list_fragment.xml

Tabel 58. Source Code list_fragment.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.constraintlayout.widget.ConstraintLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	xmlns:app="http://schemas.android.com/apk/res-auto"
5	xmlns:tools="http://schemas.android.com/tools"
6	android:layout_width="match_parent"
7	android:layout_height="match_parent"
8	tools:context="com.example.modul5.ui.GunungListFragment">
9	
10	<androidx.recyclerview.widget.RecyclerView
11	android:id="@+id/rvGunung"
12	android:layout_width="0dp"
13	android:layout_height="0dp"
14	android:clipToPadding="false"
15	android:background="#94C2EB"
16	android:padding="16dp"
17	app:layout_constraintTop_toTopOf="parent"
18	app:layout_constraintBottom_toBottomOf="parent"
19	app:layout_constraintStart_toStartOf="parent"
20	app:layout_constraintEnd_toEndOf="parent"
21	tools:listitem="@layout/item_gunung" />
22	</androidx.constraintlayout.widget.ConstraintLayout>

res/values

24. colors.xml

Tabel 59. Source Code colors.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<resources>

```

3   <color name="blue_500">#2196F3</color>
4   <color name="blue_700">#1976D2</color>
5   <color name="black">#1976D2</color>
6   <color name="blue">#1976D2</color>
7   <color name="pink">#1976D2</color>
8   <color name="teal_700">#1976D2</color>
9   <color name="cream">#FFEBBC</color>
10  <color name="white">#1976D2</color>
11  </resources>

```

25. string.xml

Tabel 60. Source Code string.xml

```

1 <resources>
2   <string name="app_name">MODUL 5</string>
3   <string name="gunung_image_desc">Gambar gunung</string>
4   <string name="btn_detail_text">Detail</string>
5   <string name="btn_favorite_text">☆</string>
6   <string name="btn_link_text">Link</string>
7 </resources>

```

res/values/themes.xml

26. themes.xml

Tabel 61. Source Code themes.xml

```

1 <resources xmlns:tools="http://schemas.android.com/tools">
2   <!-- Base application theme. -->
3   <style name="Base.Theme.Modul5"
4     parent="Theme.Material3.DayNight.NoActionBar">
5     <item name="colorPrimary">@color/blue_500</item>
6     <item name="colorOnPrimary">@android:color/white</item>
7     <item
8       name="colorPrimaryContainer">@color/blue_700</item>
9     <item
10    name="colorOnPrimaryContainer">@android:color/white</item>
11   </style>

```

12	<style name="Theme.Modul5" parent="Base.Theme.Modul5" />
13	
14	</resources>

MODUL5/app/build.gradle.kts

27. build.gradle.kts

Tabel 62. Source Code build.gradle.kts

1	plugins {
2	id("com.android.application")
3	id("org.jetbrains.kotlin.android")
4	id("kotlin-parcelize")
5	id("org.jetbrains.kotlin.plugin.serialization") version
6	"1.9.0"
7	id("org.jetbrains.kotlin.kapt")
8	}
9	
10	android {
11	namespace = "com.example.modul5"
12	compileSdk = 34
13	
14	defaultConfig {
15	applicationId = "com.example.modul5"
16	minSdk = 30
17	targetSdk = 34
18	versionCode = 1
19	versionName = "1.0"
20	
21	testInstrumentationRunner =
22	"androidx.test.runner.AndroidJUnitRunner"
23	}
24	
25	buildTypes {
26	release {
27	isMinifyEnabled = false

```
28         proguardFiles(
29             getDefaultProguardFile("proguard-android-
30 optimize.txt"),
31             "proguard-rules.pro"
32         )
33     }
34 }
35
36 compileOptions {
37     sourceCompatibility = JavaVersion.VERSION_17
38     targetCompatibility = JavaVersion.VERSION_17
39 }
40
41 kotlinOptions {
42     jvmTarget = "17"
43 }
44
45 buildFeatures {
46     viewBinding = true
47 }
48 }
49
50 dependencies {
51     implementation("androidx.core:core-ktx:1.12.0")
52     implementation("org.jetbrains.kotlin:kotlin-stdlib:1.9.0")
53     implementation("androidx.appcompat:appcompat:1.6.1")
54     implementation ("androidx.fragment:fragment-ktx:1.6.2")
55     implementation("com.google.android.material:material:1.11.0")
56
57     implementation("androidx.constraintlayout:constraintlayout:2.1.4")
58     implementation("androidx.recyclerview:recyclerview:1.3.2")
59     implementation("androidx.cardview:cardview:1.0.0")
60     implementation("androidx.lifecycle:lifecycle-runtime-
61 ktx:2.6.2")
```

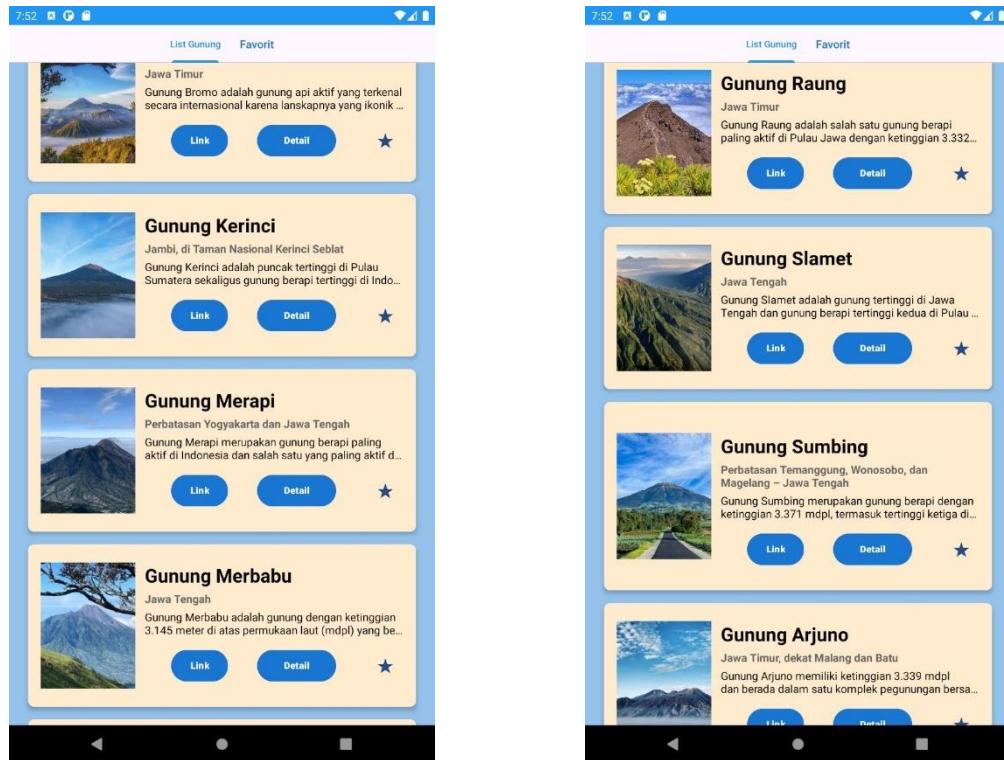
```
62     implementation("androidx.activity:activity-ktx:1.8.2")
63
64     // Retrofit & KotlinX Serialization
65     implementation("com.squareup.retrofit2:retrofit:2.9.0")
66     implementation("org.jetbrains.kotlinx:kotlinx-serialization-
67 json:1.6.0")
68     implementation("com.jakewharton.retrofit:retrofit2-kotlinx-
69 serialization-converter:0.8.0")
70
71     // **Tambahkan OkHttp Logging Interceptor supaya
72 HttpLoggingInterceptor dikenali**
73     implementation("com.squareup.okhttp3:logging-
74 interceptor:4.11.0")
75
76     // Coil for image loading
77     implementation("io.coil-kt:coil:2.5.0")
78
79     // Lifecycle & LiveData & ViewModel
80     implementation("androidx.lifecycle:lifecycle-viewmodel-
81 ktx:2.6.2")
82     implementation("androidx.lifecycle:lifecycle-livedata-
83 ktx:2.6.2")
84
85     // Room (database + coroutine support)
86     implementation("androidx.room:room-runtime:2.6.1")
87     kapt("androidx.room:room-compiler:2.6.1")
88     implementation("androidx.room:room-ktx:2.6.1")
89
90     // Kotlin Coroutines (untuk Flow)
91     implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
92 core:1.7.3")
93     implementation("org.jetbrains.kotlinx:kotlinx-coroutines-
94 android:1.7.3")
95
```

```

96     // DataStore for theme/favorite preferences
97     implementation("androidx.datastore:datastore-
98 preferences:1.0.0")
99
100    testImplementation("junit:junit:4.13.2")
101    androidTestImplementation("androidx.test.ext:junit:1.1.5")
102    androidTestImplementation("androidx.test.espresso:espresso-
103 core:3.5.1")
104 }

```

B. Output Program



7:51

http://tnungunmerbabu.org

Taman Nasional Gunung Merbabu

Home

TAMAN NASIONAL GUNUNG MERBABU

BERITABERITA.COM - TAHUN 2024 PENGUMUMAN DILAKUKAN PADA MARET 2024

dengan ketentuan sebagai berikut:

1. Jalur pendakian Sumbing dibuka kembali mulai tanggal 20 Februari 2025 dengan persentase ketinggian 50% (564 orang/jam).
2. Jalan pendakian Wates-Temanggung saat ini masih ditutup sementara sampai tahapan persetujuan pembukaan pendakian selesai dilakukan;
3. Tarif tiket wisata pendakian TN Gunung Merbabu sesuai dengan Peraturan Pemerintah Nomor 36 Tahun 2024 tentang Jenis dan Tarif atas Jenis Penerimaan Negara Bukan Pajak yang Berlaku pada Kementerian Lingkungan Hidup dan Kehutanan;
4. Pengunjung yang akan melakukan pendakian di TN Gunung Merbabu agar mematuhi Protokol Wisata Pendakian sesuai SK Kepala Balai Nomor: SK.397/35/TD/04/2024 tanggal 26 April 2024;

Pembukaan Objek Wisata Pendakian Taman Nasional Gunung Merbabu

Posted by: admin February 17, 2025

Aksi Bersih Gunung di Jalur Pendakian Gunung Merbabu Thekelan

Dalam rangka memperingati Hari Peduli Sampah Nasional (HPSN) tahun 2025, Balai TN Gunung Merbabu ...

Posted by: admin February 16, 2025

Wisata Edukasi dan Konservasi di Taman Nasional Gunung Merbabu, berupa Pengamatan Burung (Birdwatching) di Hutan Bratanya

Potensi jenis burung di TNGMb Keaneragaman burung di Taman Nasional Gunung Merbabu (TNGMb) termasuk ...

Posted by: admin December 24, 2024

6:53

List Gunung Favorit

Gunung Sumbing

Perbatasan Temanggung, Wonosobo, dan Magelang – Jawa Tengah

Gunung Sumbing merupakan gunung berapi dengan ketinggian 3.371 mdpl, merupakan tertinggi ketiga di Pulau Jawa. Gunung ini berdampingan erat dengan Gunung Sindoro, menciptakan panorama ‘kembang’ yang sangat populer di kalangan pendaki dan fotografer. Jalur pendakian populer antara lain via Garung (Wonosobo), Bowongso, dan Banaran (Temanggung). Di dekat puncaknya terdapat kawah aktif kecil yang terkadang mengeluarkan asap belerang. Vegetasi yang dilalui meliputi hutan tropis, ladang pertanian warga, hingga jalur berbatu dan padang rumput yang luas. Puncak Sumbing menawarkan pemandangan spektakuler ke gunung-gunung di sekitarnya, termasuk Merapi, Merbabu, Sindoro, dan Slamet. Meskipun tergolong aktif, aktivitas vulkanik di Sumbing relatif jarang terjadi dalam skala besar.

Kembali

6:51

List Gunung

Gunung Bromo

Jawa Timur

Gunung Bromo adalah gunung api aktif yang terkenal secara internasional karena lanskapnya yang ikonik ...

Link Detail

Gunung Kerinci

Jambi, di Taman Nasional Kerinci Seblat

Gunung Kerinci adalah puncak tertinggi di Pulau Sumatera sekaligus gunung berapi tertinggi di Indo...

Link Detail

Gunung Merapi

Perbatasan Yogyakarta dan Jawa Tengah

Gunung Merapi merupakan gunung berapi paling aktif di Indonesia dan salah satu yang paling aktif d...

Link Detail

Gunung Merbabu

Jawa Tengah

Gunung Merbabu adalah gunung dengan ketinggian 3.145 meter di atas permukaan laut (mdpl) yang ber...

Link Detail

6:51

List Gunung Favorit

Gunung Bromo

Jawa Timur

Gunung Bromo adalah gunung api aktif yang terkenal secara internasional karena lanskapnya yang ikonik dan ...

Link Detail

Gunung Kerinci

Jambi, di Taman Nasional Kerinci Seblat

Gunung Kerinci adalah puncak tertinggi di Pulau Sumatera sekaligus gunung berapi tertinggi di Indonesia dengan to...

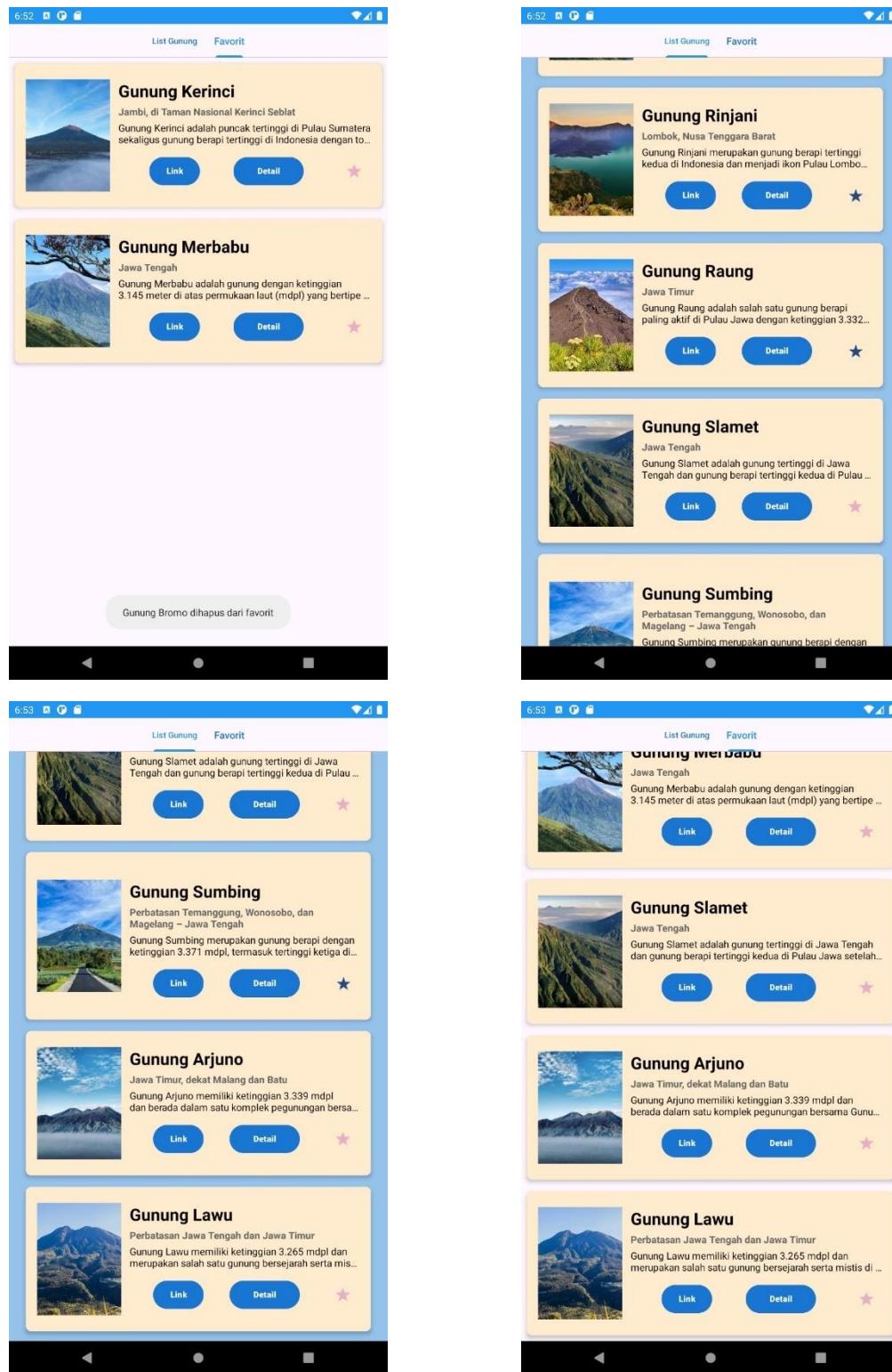
Link Detail

Gunung Merapi

Jawa Tengah

Gunung Merapi merupakan gunung berapi tertinggi di Indonesia dengan ketinggian 3.145 meter di atas permukaan laut (mdpl) yang ber...

Link Detail



Gambar 9. Screeenshot Output Program Soal 1

C. Pembahasan

MODUL5/app/src/main/AndroidManifest.xml

1. AndroidManifest.xml

Kode XML dikelas ini adalah file AndroidManifest.xml, yaitu file penting dalam setiap proyek Android yang berfungsi untuk mendeklarasikan informasi dasar aplikasi kepada sistem Android. File dimulai dengan deklarasi XML standar <?xml version="1.0" encoding="utf-8"?>, yang menunjukkan bahwa dokumen ini ditulis dalam format XML dengan encoding UTF-8. Tag <manifest> menjadi elemen utama dan wajib di setiap file manifest. Di dalamnya, terdapat atribut package="com.example.modul5" yang mendefinisikan nama paket unik dari aplikasi ini, yang juga menjadi identitas aplikasi di Play Store maupun sistem Android. Selanjutnya, ada dua deklarasi permission (<uses-permission>) yang memberi tahu sistem bahwa aplikasi membutuhkan izin tertentu. Yang pertama adalah android.permission.ACCESS_NETWORK_STATE, yang memungkinkan aplikasi memeriksa status jaringan (apakah terhubung ke WiFi, mobile data, atau offline). Yang kedua adalah android.permission.INTERNET, yang memberi aplikasi akses untuk terhubung ke internet, penting untuk aplikasi yang mengambil data dari API atau layanan online. Di dalam tag <application>, berbagai atribut disetel untuk mengatur properti aplikasi secara umum. Atribut android:allowBackup="true" mengizinkan pengguna untuk membackup data aplikasi ke akun Google mereka.

Atribut android:dataExtractionRules dan android:fullBackupContent merujuk ke file XML eksternal yang mengatur aturan ekstraksi data dan konten yang diikutsertakan saat backup. Atribut visual seperti android:icon dan android:roundIcon menetapkan ikon aplikasi dalam bentuk standar dan bulat. android:label mengambil nilai dari strings.xml untuk memberi nama aplikasi yang akan ditampilkan ke pengguna. android:supportsRtl="true" menunjukkan bahwa aplikasi ini mendukung layout dari kanan ke kiri, penting untuk bahasa-bahasa seperti Arab atau Ibrani. android:theme menentukan tema global aplikasi yang diambil dari file style (Theme.Modul5). Di dalam application, terdapat deklarasi sebuah activity, yakni MainActivity, yang merupakan pintu masuk utama aplikasi. Atribut android:exported="true" diperlukan untuk menjelaskan apakah activity ini bisa diakses dari

luar aplikasi, dan sejak Android 12, atribut ini wajib dideklarasikan untuk komponen yang memiliki intent-filter. Di dalam tag <intent-filter>, terdapat dua elemen penting: <action android:name="android.intent.action.MAIN" /> dan <category android:name="android.intent.category.LAUNCHER" />. Kombinasi keduanya membuat MainActivity menjadi activity pertama yang dijalankan saat aplikasi dibuka dari launcher. Secara keseluruhan, file ini mendefinisikan hak akses, tampilan, aktivitas utama, serta dukungan backup untuk aplikasi Android yang bernama Modul5. Semua komponen ini saling berkaitan untuk menjamin aplikasi berjalan dengan baik sesuai standar sistem operasi Android.

com/example/modul5/data/local/entity/dao

2. GunungDao

Kode Kotlin di kelas ini mendefinisikan sebuah **interface bernama GunungDao** (Data Access Object) yang berfungsi sebagai penghubung antara aplikasi dengan database lokal yang dibangun menggunakan **Room**, yaitu library ORM (Object Relational Mapping) pada Android. Interface ini menangani segala operasi database terhadap entitas GunungEntity, yaitu representasi dari data gunung yang tersimpan di tabel gunung. Seluruh fungsi yang ada di dalam GunungDao dibuat untuk menangani pengambilan, penyisipan, penghapusan, dan pembaruan data, baik secara sinkron maupun reaktif. Interface ini diawali dengan anotasi `@Dao` yang menandakan bahwa interface ini adalah komponen DAO. Room akan secara otomatis menghasilkan implementasi konkret dari interface ini saat build time. Fungsi pertama adalah `getAllGunung()`, yang menggunakan anotasi `@Query("SELECT * FROM gunung")` untuk mengambil seluruh data dari tabel gunung. Fungsi ini mengembalikan data dalam bentuk `Flow<List<GunungEntity>>`, yang artinya data tersebut bisa *di-observe* secara real-time. Jika ada perubahan di tabel, maka data yang diterima UI juga akan otomatis terbarui.

Fungsi kedua, `getAllGunungOnce()`, juga mengambil seluruh data dari tabel gunung, tetapi kali ini menggunakan tipe `suspend` yang artinya hanya dieksekusi sekali dalam coroutine dan hasilnya dikembalikan sebagai `List<GunungEntity>`. Tidak seperti `Flow`, data ini tidak bersifat reaktif. Selanjutnya, `getAllFavoriteGunung()` adalah fungsi yang

mengambil data gunung yang berstatus favorit, ditandai dengan `isFavorite = 1`. Sama seperti sebelumnya, fungsi ini bersifat satu kali eksekusi (suspend), dan hasilnya berupa daftar objek `GunungEntity` favorit. Sebaliknya, `getAllFavoriteGunungFlow()` mengambil data favorit dengan cara yang reaktif menggunakan Flow. Ini berguna untuk UI yang ingin selalu memperbarui tampilan berdasarkan perubahan status favorit. Fungsi `insertGunungList()` bertugas untuk menyimpan banyak data gunung sekaligus ke dalam database. Ia menggunakan strategi konflik REPLACE, yang berarti jika ada data dengan primary key yang sama, data lama akan ditimpas dengan yang baru. Demikian pula, `insertGunung()` melakukan hal yang sama seperti fungsi sebelumnya, tetapi hanya untuk satu entitas `GunungEntity`. Fungsi `clearGunung()` adalah perintah untuk menghapus seluruh data dalam tabel gunung. Ini berguna saat aplikasi ingin me-reset database lokal, misalnya setelah logout atau sinkronisasi ulang data. Fungsi `delete(gunung: GunungEntity)` digunakan untuk menghapus satu entitas gunung tertentu. Room akan mencari berdasarkan primary key dari objek tersebut lalu menghapus datanya dari tabel.

Kemudian ada fungsi `update(gunung: GunungEntity)` yang akan memperbarui seluruh isi data gunung berdasarkan primary key-nya. Fungsi ini berguna saat ingin mengganti informasi seperti deskripsi atau status gunung. Terakhir, `updateFavoriteStatus(name: String, isFavorite: Boolean)` memungkinkan perubahan nilai kolom `isFavorite` secara spesifik berdasarkan nama gunung. Ini sangat efisien saat hanya ingin memperbarui status favorit tanpa harus mengubah keseluruhan entitas. Secara keseluruhan, `GunungDao` menyediakan fungsi-fungsi yang lengkap untuk menangani seluruh siklus data gunung di aplikasi mulai dari membaca, menyisipkan, memperbarui, hingga menghapus data. Kombinasi antara penggunaan suspend untuk operasi sekali jalan dan Flow untuk data reaktif menunjukkan praktik modern dalam pengelolaan database pada aplikasi Android berbasis Kotlin.

[com/example/modul5/data/local/entity/database](#)

3. AppDatabase

Kode yang ditampilkan adalah bagian penting dari implementasi database lokal pada aplikasi Android menggunakan library Room. Secara umum, kode ini membentuk kerangka

kerja utama untuk mengatur penyimpanan data dalam database SQLite dengan pendekatan yang lebih terstruktur, aman, dan mudah diakses melalui konsep object-oriented. File ini terletak dalam package com.example.modul5.data.local.database dan berisi deklarasi kelas abstrak AppDatabase yang merupakan turunan dari RoomDatabase. Kelas ini menjadi representasi langsung dari database itu sendiri dan harus ditandai dengan anotasi @Database. Di dalam anotasi ini, ditentukan entitas-entitas yang digunakan dalam database, yakni kelas-kelas data yang mewakili tabel. Dalam hal ini, hanya terdapat satu entitas, yaitu GunungEntity, yang kemungkinan besar menyimpan data tentang gunung-gunung, seperti nama, lokasi, ketinggian, dan status favorit. Selain itu, database ini diberi versi pertama (version = 1) sebagai penanda bahwa ini adalah struktur awal dari database yang dibuat.

Sementara itu, properti exportSchema diset ke false, artinya Room tidak akan mengekspor skema database ke file JSON—biasanya pengaturan ini digunakan ketika skema tidak diperlukan untuk dokumentasi atau pengujian. Kelas AppDatabase mendefinisikan satu fungsi abstrak, gunungDao(), yang mengembalikan sebuah objek GunungDao. DAO (Data Access Object) adalah antarmuka yang menyediakan berbagai metode untuk berinteraksi dengan tabel gunung dalam database, seperti membaca semua data, menambahkan, memperbarui, atau menghapus data. Dengan pendekatan ini, semua logika query disederhanakan dan dipisahkan dari logika tampilan, sehingga mendukung arsitektur aplikasi yang bersih dan terorganisir. Yang paling krusial dari kode ini adalah bagian companion object, yaitu sebuah objek statis yang memungkinkan akses ke metode dan properti tanpa harus membuat instance dari kelas AppDatabase. Di dalamnya terdapat variabel INSTANCE, yang ditandai dengan anotasi @Volatile. Anotasi ini memastikan bahwa perubahan pada variabel tersebut akan langsung terlihat oleh thread lain, sehingga menjaga konsistensi dan mencegah kondisi tidak stabil saat beberapa thread mencoba mengakses database secara bersamaan. Fungsi getDatabase(context: Context) di dalam companion object ini bertugas mengelola pembuatan dan pengembalian instance tunggal dari database. Fungsi ini menggunakan teknik *lazy initialization*, di mana instance database hanya akan dibuat saat pertama kali dibutuhkan. Mekanisme synchronized(this) digunakan agar hanya satu thread yang dapat mengeksekusi bagian ini pada satu waktu, sehingga menghindari duplikasi pembuatan database yang bisa menimbulkan error atau penggunaan sumber daya yang tidak efisien. Jika INSTANCE masih null, maka Room.databaseBuilder() akan dipanggil untuk

membuat database baru, menggunakan applicationContext dari context yang diberikan agar tidak terjadi memory leak akibat penggunaan context dari activity atau fragment.

Nama database yang digunakan di sini adalah "gunung_database", yang berarti Room akan membuat file SQLite dengan nama tersebut di penyimpanan internal aplikasi. Setelah instance database berhasil dibuat, ia disimpan ke dalam variabel INSTANCE agar dapat digunakan kembali di masa mendatang tanpa harus membuat ulang. Fungsi ini lalu mengembalikan instance tersebut kepada pemanggil. Secara keseluruhan, kode ini adalah struktur penting yang memastikan bahwa aplikasi memiliki satu akses pusat yang aman, efisien, dan terstruktur terhadap database lokal. Ia mengatur cara aplikasi menyimpan, mengambil, dan mengelola data gunung melalui GunungDao, dan menjamin bahwa interaksi dengan database dilakukan melalui satu pintu dengan menggunakan pola singleton. Pendekatan ini sangat sesuai untuk arsitektur modern berbasis MVVM atau Clean Architecture di Android.

com/example/modul5/data/local/entity/entity

4. GunungEntity.kt

Kode ini merupakan bagian dari sistem database lokal menggunakan **Room** di aplikasi Android, dan berfungsi untuk mendefinisikan struktur data atau *entity* yang akan disimpan di dalam tabel database. File ini berada dalam package com.example.modul5.data.local.entity, yang berarti fungsinya terfokus pada mendefinisikan *entitas* yang mewakili satu tabel bernama gunung dalam database lokal aplikasi. Kelas yang dideklarasikan di sini adalah GunungEntity, dan ditandai dengan anotasi @Entity(tableName = "gunung"). Anotasi ini memberi tahu Room bahwa kelas GunungEntity adalah representasi dari sebuah tabel database bernama gunung. Dengan kata lain, setiap objek GunungEntity akan merepresentasikan satu baris data di dalam tabel tersebut.

Kelas ini menggunakan data class, yang merupakan fitur Kotlin untuk membuat kelas yang berfungsi menyimpan data dengan boilerplate code yang minimal. Di dalamnya terdapat beberapa properti atau kolom, yaitu:

- `@PrimaryKey val name: String`: Kolom name berfungsi sebagai *primary key*, yang berarti nilainya harus unik dan digunakan Room untuk mengidentifikasi tiap entri

secara individual. Dalam konteks ini, name kemungkinan merupakan nama gunung dan digunakan sebagai identitas utama tiap data.

- val lokasi: String: Kolom lokasi menyimpan informasi lokasi gunung, seperti nama provinsi, kabupaten, atau area geografis tempat gunung berada.
- val deskripsi: String: Kolom ini menyimpan deskripsi tentang gunung, yang bisa berupa informasi sejarah, karakteristik, keindahan, atau daya tariknya.
- val link: String: Merupakan kolom yang berisi tautan, kemungkinan ke sumber informasi lebih lanjut seperti artikel, blog, atau situs resmi yang memuat informasi mendetail tentang gunung tersebut.
- val image: String: Kolom ini menyimpan URL atau path gambar gunung, yang akan digunakan oleh aplikasi untuk menampilkan visualisasi gunung di antarmuka pengguna.
- val isFavorite: Boolean: Kolom ini berfungsi sebagai penanda apakah gunung tersebut termasuk dalam daftar favorit pengguna atau tidak. Nilainya berupa true atau false, dan bisa digunakan untuk fitur filter atau penyimpanan personalisasi data.

Dengan struktur ini, Room dapat secara otomatis menghasilkan kode SQL untuk membuat tabel gunung dan mengelola datanya tanpa harus menulis query secara manual dalam setiap operasi. Setiap kali aplikasi menyimpan atau mengambil data dari tabel gunung, ia akan berurusan dengan objek GunungEntity. Pendekatan ini mendukung konsep *type-safe*, yang meminimalkan kesalahan saat berinteraksi dengan database karena semua struktur dan tipe data sudah didefinisikan secara eksplisit di kelas ini. Kode ini juga merupakan bagian penting dalam arsitektur berbasis MVVM (Model-View-ViewModel) karena menjadi lapisan Model yang menyimpan dan menyediakan data secara terstruktur dan konsisten ke seluruh bagian aplikasi.

<com/example/modul5/data/mapper>

5. GunungMapper.kt

Kode ini berada di dalam package com.example.modul5.data.mapper, yang menunjukkan bahwa fungsinya adalah sebagai *mapper*, atau pengubah data, antara dua

lapisan data berbeda dalam arsitektur aplikasi: yaitu model data dari sumber eksternal (Gunung) dan entitas database lokal (GunungEntity). Mapper ini sangat penting untuk menjaga pemisahan tanggung jawab antar lapisan, terutama dalam arsitektur MVVM atau Clean Architecture, di mana data dari dan ke database perlu dipisahkan dari data yang digunakan di layer domain atau UI. Di dalam file ini terdapat dua fungsi ekstensi Kotlin, yang memungkinkan penulisan metode seolah-olah menjadi bagian dari kelas aslinya, meskipun dideklarasikan secara eksternal. Fungsi pertama adalah fun Gunung.toEntity(): GunungEntity. Fungsi ini adalah ekstensi dari kelas Gunung, yang berarti fungsi ini hanya bisa dipanggil dari objek bertipe Gunung. Tujuannya adalah untuk mengonversi objek Gunung menjadi objek GunungEntity.

Fungsi ini digunakan ketika data dari sumber eksternal (seperti API atau sumber data lainnya) ingin disimpan ke dalam database lokal Room. Semua properti seperti name, lokasi, deskripsi, link, image, dan isFavorite dipetakan secara langsung dari objek Gunung ke GunungEntity. Hal ini memastikan bahwa data dapat disimpan di tabel database gunung dengan struktur yang sesuai. Fungsi kedua adalah fun GunungEntity.toModel(): Gunung. Kebalikan dari fungsi sebelumnya, fungsi ini merupakan ekstensi dari kelas GunungEntity, dan digunakan untuk mengonversi data dari entitas database menjadi model Gunung yang bisa digunakan di domain aplikasi atau ditampilkan di UI. Proses konversi ini penting karena entitas Room biasanya digunakan hanya di lapisan data atau repository, dan tidak langsung digunakan oleh UI. Dengan fungsi ini, data yang sebelumnya tersimpan dalam database dapat diubah ke bentuk yang lebih netral dan sesuai kebutuhan tampilan aplikasi.

Kedua fungsi ini, meskipun tampak sederhana, memiliki peran penting dalam menjaga fleksibilitas dan skalabilitas aplikasi. Dengan memisahkan struktur data yang digunakan untuk menyimpan (entitas) dan struktur data yang digunakan untuk berinteraksi dengan UI atau logika aplikasi (model), maka perubahan pada satu sisi tidak langsung memengaruhi sisi lainnya. Misalnya, jika struktur database perlu diperbarui atau jika model Gunung ingin ditambahkan atribut baru khusus untuk tampilan, perubahan tersebut bisa dilakukan tanpa mengganggu keseluruhan sistem, cukup dengan menyesuaikan fungsi mapper ini. Dengan begitu, kode menjadi lebih modular, mudah diuji, dan lebih tahan terhadap perubahan.

com/example/modul5/data/model

6. Gunung

Kode ini berada di dalam package com.example.modul5.data.model, dan mendefinisikan sebuah *data class* bernama Gunung yang merupakan representasi dari model data utama dalam aplikasi. Kelas ini menampung informasi detail tentang gunung, dan digunakan untuk menyimpan serta memproses data dari sumber eksternal seperti API sebelum ditampilkan ke pengguna atau diolah lebih lanjut dalam aplikasi. Kata kunci `@Serializable` yang ada di atas kelas menunjukkan bahwa Gunung adalah kelas yang dapat deserialisasi menggunakan pustaka `kotlinx.serialization`. Artinya, objek dari kelas ini dapat diubah menjadi format lain seperti JSON dan sebaliknya. Ini sangat penting dalam komunikasi client-server, karena data dari API umumnya datang dalam format JSON dan harus dikonversi ke objek Kotlin untuk dapat digunakan dalam kode aplikasi. Setiap properti di dalam kelas Gunung diberi anotasi `@SerializedName`, yang menunjukkan nama sebenarnya dari properti tersebut seperti yang muncul di JSON dari API. Ini memungkinkan Kotlin untuk memetakan field JSON ke nama properti Kotlin yang digunakan di aplikasi. Misalnya:

- `@SerializedName("name") val name: String` menunjukkan bahwa JSON memiliki field "name" yang akan dimasukkan ke properti name.
- `@SerializedName("lokasi") val lokasi: String` memetakan field "lokasi" dari JSON ke properti lokasi.
- `@SerializedName("deskripsi") val deskripsi: String` untuk isi deskripsi gunung.
- `@SerializedName("link") val link: String` yang biasanya merujuk ke tautan sumber atau informasi tambahan.
- `@SerializedName("image_url") val image: String` menunjukkan bahwa data gambar gunung dikirim melalui field JSON bernama "image_url", dan disimpan di properti image.

Satu properti, yaitu `isFavorite`, tidak memiliki anotasi `@SerializedName`, karena diasumsikan bahwa field ini tidak berasal dari JSON API, melainkan properti tambahan yang digunakan di dalam aplikasi itu sendiri, misalnya untuk menandai gunung favorit. Nilai default dari

properti ini adalah false, sehingga ketika objek Gunung dibuat dari JSON dan field isFavorite tidak ditemukan, nilainya otomatis akan false. Di akhir deklarasi kelas, terdapat pewarisan dari JavaSerializable (java.io.Serializable) yang diberi alias agar tidak konflik dengan anotasi KotlinX. Ini berguna ketika objek Gunung perlu dipindahkan antar komponen Android, seperti dari satu fragment ke fragment lain melalui Bundle atau Intent, karena Android memerlukan objek yang bisa diserialisasi dalam proses itu. Secara keseluruhan, kelas Gunung ini merupakan inti dari model data aplikasi, yang merepresentasikan struktur data gunung dari API, mendukung konversi JSON, dan juga siap digunakan dalam konteks navigasi Android. Pendekatan ini menjadikan data model fleksibel untuk kebutuhan backend (API), frontend (UI), maupun penyimpanan sementara (seperti pada Bundle).

com/example/modul5/data/remote

7. ApiClient

Kode di atas berada dalam package com.example.modul5.data.remote dan berfungsi sebagai konfigurasi client Retrofit untuk komunikasi dengan API eksternal. Seluruh konfigurasi ini dibungkus dalam sebuah object Kotlin bernama ApiClient, yang berarti kelas ini bersifat singleton—hanya akan dibuat satu instance selama aplikasi berjalan. Tujuan utama dari ApiClient adalah untuk mengatur bagaimana aplikasi akan melakukan request HTTP ke server, dalam hal ini ke alamat <https://modul5.free.beeceptor.com/api/path/>. Bagian pertama adalah konstanta BASE_URL, yaitu URL dasar yang akan menjadi prefix dari semua endpoint yang digunakan untuk request data gunung dari API. Selanjutnya, disiapkan konfigurasi Json dari kotlinc.serialization.json. Di sini properti ignoreUnknownKeys = true digunakan agar saat melakukan deserialisasi dari JSON ke objek Kotlin, field-field yang tidak dikenali oleh data class tidak menyebabkan error. Hal ini sangat berguna ketika struktur JSON dari API memiliki elemen tambahan yang tidak digunakan oleh aplikasi.

Kemudian ada loggingInterceptor, sebuah instance dari HttpLoggingInterceptor dari pustaka OkHttp, yang diatur pada level BODY. Ini artinya semua detail dari request dan response, termasuk header dan body, akan dicetak ke log. Logging ini sangat membantu saat proses debugging atau pengujian karena developer bisa melihat isi sebenarnya dari

komunikasi HTTP yang sedang berlangsung. Setelah itu, okHttpClient dikonfigurasi menggunakan builder pattern. Di dalamnya, loggingInterceptor ditambahkan agar setiap request dan response bisa tercatat. Timeout untuk koneksi dan pembacaan data juga diatur selama 30 detik agar koneksi tidak langsung gagal saat jaringan lambat. Selanjutnya dibuat instance retrofit menggunakan builder milik Retrofit. Di sini ditetapkan baseUrl dan client yang telah dikonfigurasi sebelumnya. Selain itu, konversi dari JSON ke objek Kotlin dilakukan dengan addConverterFactory, yang memanfaatkan pustaka kotlinx.serialization dan diubah menjadi converter Retrofit menggunakan asConverterFactory. Format konten yang digunakan untuk konversi adalah application/json, yang merupakan standar umum untuk komunikasi data di web API. Di bagian akhir, gunung ApiService dideklarasikan sebagai objek service Retrofit yang dibuat dari interface Gunung ApiService. Interface ini akan berisi definisi endpoint dan metode-metode HTTP seperti GET, POST, atau lainnya yang digunakan untuk mengambil data gunung dari API. Secara keseluruhan, ApiClient ini merupakan komponen utama yang memungkinkan aplikasi untuk berkomunikasi dengan API secara efisien dan aman. Ia memanfaatkan Retrofit dan OkHttp sebagai alat komunikasi, serta kotlinx.serialization sebagai alat konversi data. Kombinasi ini menjadikan aplikasi tidak hanya modular dan mudah di-maintain, tapi juga cukup fleksibel untuk berinteraksi dengan berbagai format data dari server.

8. Gunung ApiService

Kode di atas merupakan sebuah interface bernama Gunung ApiService yang berada di dalam package com.example.modul5.data.remote. Interface ini digunakan sebagai kontrak API untuk komunikasi antara aplikasi dan server eksternal menggunakan Retrofit, sebuah library populer di Android untuk melakukan HTTP request secara efisien. Pada dasarnya, interface ini mendefinisikan endpoint yang dapat dipanggil oleh aplikasi untuk mendapatkan data gunung dari server. Di dalam interface ini terdapat satu fungsi, yaitu getGunungList(). Fungsi getGunungList() diberi anotasi @GET("gunung"), yang berarti saat fungsi ini dipanggil, Retrofit akan melakukan permintaan HTTP GET ke endpoint yang bernama "gunung", yang merupakan bagian dari URL lengkap yang sudah didefinisikan sebelumnya dalam ApiClient melalui properti BASE_URL. Jadi, URL lengkapnya menjadi <https://modul5.free.beeceptor.com/api/path/gunung>.

Fungsi ini bersifat suspend, yang artinya hanya bisa dipanggil dari dalam coroutine atau fungsi lain yang juga bersifat suspend. Ini memungkinkan fungsi untuk berjalan secara asynchronous (non-blocking), yang penting dalam pengembangan aplikasi Android agar tidak membekukan UI saat mengambil data dari internet. Hasil dari fungsi ini adalah `List<Gunung>`, yakni daftar objek `Gunung`, yang merupakan data class yang telah dibuat sebelumnya dan merepresentasikan model gunung dengan informasi seperti nama, lokasi, deskripsi, link, gambar, dan status favorit. Dengan kata lain, `Gunung ApiService` ini adalah penghubung antara aplikasi dan API eksternal. Ia mendeskripsikan bagaimana aplikasi dapat meminta data dari internet (dalam hal ini, daftar gunung), dan Retrofit akan secara otomatis menangani proses permintaan HTTP, parsing JSON ke objek Kotlin (`Gunung`), serta pengelolaan threading melalui coroutine. Struktur seperti ini membuat kode menjadi bersih, mudah diuji, dan sangat terintegrasi dengan arsitektur modern Android.

`com/example/modul5/data/repository`

9. GunungRepository

Kode tersebut merupakan implementasi dari sebuah *repository class* bernama `GunungRepository`, yang memiliki peran penting dalam arsitektur aplikasi modern seperti MVVM (Model-View-ViewModel). Kelas ini mengatur alur data dari dua sumber utama—API (melalui internet) dan database lokal (melalui Room)—dan menyajikannya ke `ViewModel` secara bersih dan terstruktur. Kelas ini juga bertanggung jawab mengelola transformasi data dan sinkronisasi antara dua sumber data tersebut. Pada bagian atas kelas, terdapat inisialisasi dua properti penting: `apiService` dan `gunungDao`. `apiService` merupakan instance dari `Gunung ApiService` yang didapat dari singleton `ApiClient`, dan digunakan untuk mengambil data dari internet. Sedangkan `gunungDao` diperoleh dari `AppDatabase` dengan memanggil fungsi `getDatabase(context)` dan mengambil `gunungDao()`-nya, yang berfungsi untuk akses ke database lokal. Salah satu fungsi utama dalam repository ini adalah `getFavoriteGunungListFlow()`. Fungsi ini mengembalikan data daftar gunung favorit dalam bentuk *Flow*, yang mendukung pengamatan data secara reaktif.

Data yang diambil dari database berupa `List<GunungEntity>`, lalu diubah menjadi `List<Gunung>` dengan menggunakan map dan ekstensi `toModel()` dari package mapper. Ini memungkinkan UI untuk otomatis menerima pembaruan jika data gunung favorit berubah di

database. Fungsi yang paling kompleks di dalam kelas ini adalah `fetchGunungList()`. Fungsi ini bekerja dalam coroutine dengan konteks `Dispatchers.IO` untuk operasi I/O, yang artinya dijalankan di thread terpisah agar tidak mengganggu thread utama. Di dalamnya, pertama-tama diambil data gunung dari server melalui `apiService.getGunungList()`. Setelah itu, daftar nama gunung favorit dari database lokal diambil untuk melakukan pencocokan. Daftar gunung dari API kemudian *di-merge*—jika nama gunung cocok dengan salah satu nama favorit, maka nilai `isFavorite` akan disetel ke `true`. Ini memastikan data favorit tetap sinkron setelah mengambil data baru dari server. Setelah penggabungan selesai, database lokal dibersihkan (`clearGunung()`), lalu data baru yang telah digabung diubah menjadi entitas dan disimpan kembali (`insertGunungList()`). Jika terjadi error saat pengambilan data dari internet (misalnya karena tidak ada koneksi), fungsi akan mengambil data dari database lokal sebagai *fallback*, dan mengonversinya ke model `Gunung`. Selain itu, terdapat tiga fungsi lainnya: `updateGunung()`, `insertFavoriteGunung()`, dan `deleteFavoriteGunung()`. Ketiganya digunakan untuk memperbarui data gunung tertentu di database.

`updateGunung()` digunakan untuk memperbarui seluruh entitas gunung, sedangkan dua fungsi lainnya hanya memodifikasi status favorit sebuah gunung berdasarkan nama—menandainya sebagai favorit (`true`) atau menghapusnya dari favorit (`false`). Secara keseluruhan, `GunungRepository` ini menjadi lapisan penghubung antara `ViewModel` dengan sumber data, menyatukan logika akses data dari dua arah (API dan database lokal), serta mengatur bagaimana data diolah dan disajikan agar tetap konsisten dan efisien. Pendekatan ini membuat kode aplikasi lebih mudah dirawat, diuji, dan dikembangkan secara modular.

10. RepositoryInstance

Kode di atas mendefinisikan sebuah *singleton object* bernama `RepositoryInstance` yang digunakan untuk menyediakan instance tunggal dari kelas `GunungRepository`. Tujuan utama dari pendekatan ini adalah untuk menjamin bahwa hanya satu instance `GunungRepository` yang digunakan sepanjang siklus hidup aplikasi, sehingga efisiensi memori terjaga dan konsistensi data tetap stabil. Di dalam `RepositoryInstance`, terdapat sebuah properti `repository` bertipe nullable `GunungRepository`, yang diset null secara default. Properti ini menjadi tempat penyimpanan instance repository yang sesungguhnya. Fungsi inti

dari objek ini adalah `provideRepository(context: Context)`. Fungsi ini akan mengembalikan instance `GunungRepository` yang aktif. Namun sebelum memberikan instance tersebut, fungsi akan memeriksa apakah repository sudah pernah dibuat sebelumnya.

Jika belum (masih null), maka blok `synchronized(this)` akan dijalankan untuk memastikan bahwa hanya satu thread yang dapat membuat instance pada satu waktu—mencegah kondisi race atau *double instantiation* dalam lingkungan multi-thread. Di dalam blok `synchronized`, objek `GunungRepository` dibuat dengan menyuplai `ApplicationContext` dari konteks yang diterima sebagai argumen. Penggunaan `ApplicationContext` di sini sangat penting untuk mencegah potensi *memory leak*, karena `ApplicationContext` memiliki siklus hidup sepanjang aplikasi berjalan, bukan hanya sepanjang siklus hidup activity atau fragment. Setelah instance berhasil dibuat, ia disimpan dalam variabel `repository` agar bisa digunakan kembali saat fungsi `provideRepository()` dipanggil lagi di kemudian hari. Fungsi lalu mengembalikan instance tersebut, baik itu instance baru maupun instance yang sudah ada sebelumnya. Dengan pendekatan seperti ini, `RepositoryInstance` memberikan cara yang aman, efisien, dan terpusat dalam menyediakan akses ke `GunungRepository`, yang sangat berguna terutama dalam arsitektur MVVM, di mana `ViewModel` membutuhkan sumber data yang konsisten dari satu repository utama. Pendekatan ini juga sangat cocok diterapkan dalam aplikasi yang mengandalkan dependency injection manual tanpa menggunakan framework seperti Hilt atau Dagger.

[com/example/modul5/ui](https://github.com/putriyunita/TravelApp/blob/main/app/com/example/modul5/ui)

11. FavoriteGunungFragment.kt

Kode di atas mendefinisikan kelas `FavoritGunungFragment`, yang merupakan salah satu bagian dari antarmuka pengguna dalam aplikasi Android berbasis arsitektur MVVM. Fragment ini bertugas menampilkan daftar gunung yang telah ditandai sebagai favorit oleh pengguna. Ia memanfaatkan `ViewModel`, `RecyclerView`, dan `ViewBinding` untuk mengelola data dan tampilan secara efisien. Dalam deklarasi kelas, terdapat properti `_binding` yang merupakan instance dari `FragmentFavoritGunungBinding`, yaitu binding yang dibuat otomatis berdasarkan file XML `fragment_favorit_gunung.xml`. Binding ini digunakan untuk

mengakses elemen UI tanpa harus memanggil `findViewById`, yang membuat kode lebih ringkas dan aman dari kesalahan penamaan. Binding hanya aktif selama fragment memiliki view, dan diatur menjadi null kembali di `onDestroyView()` untuk mencegah memory leak. Fragment ini menggunakan pola by `viewModels` untuk menginisialisasi `GunungViewModel`.

ViewModel disediakan melalui `GunungViewModelFactory`, yang mengambil instance repository dari `RepositoryInstance.provideRepository()` dengan menggunakan `requireContext()` untuk mendapatkan context aplikasi. Ini memastikan bahwa ViewModel memiliki akses ke sumber data yang konsisten. Pada `onCreateView`, layout fragment di *inflate* dan view root-nya dikembalikan. Lalu di `onViewCreated`, adapter dari `RecyclerView` diatur menggunakan kelas `GunungAdapter`. Adapter ini menerima tiga parameter fungsi lambda: satu untuk membuka link gunung di browser, satu untuk menampilkan detail gunung di fragment baru, dan satu lagi untuk mengubah status favorit gunung. Setiap aksi direspon dengan membuat intent (untuk membuka link), memanggil `FragmentTransaction` (untuk navigasi ke detail), atau memperbarui data melalui `viewModel.updateGunung()`. `RecyclerView` disiapkan dengan `LinearLayoutManager` agar item ditampilkan dalam daftar vertikal, dan adapter yang telah dikonfigurasi ditetapkan sebagai adapter-nya. Untuk memuat daftar favorit secara dinamis, fragment menggunakan `lifecycleScope.launch` yang menjalankan coroutine untuk mengoleksi aliran data `favoriteList` dari ViewModel.

Setiap kali data favorit berubah, adapter akan diperbarui dengan data terbaru melalui `submitList()`. Terakhir, `onDestroyView` bertugas untuk menghapus referensi ke binding ketika fragment dihancurkan, menjaga manajemen memori tetap efisien dan mencegah kebocoran memori karena view yang tidak lagi aktif tetap tertahan di memori. Keseluruhan implementasi ini menunjukkan pendekatan yang bersih dan terstruktur untuk menampilkan daftar data yang reaktif di Android, sambil menjaga keterpisahan logika tampilan dan logika data.

12. GunungAdapter

Kode di atas mendefinisikan `GunungAdapter`, yaitu kelas adapter untuk `RecyclerView` yang digunakan dalam aplikasi Android. Adapter ini bertugas mengatur bagaimana data gunung ditampilkan dalam bentuk daftar di antarmuka pengguna. Kelas ini

mewarisi dari `ListAdapter<Gunung, GunungAdapter.ListViewHolder>`, yang membuatnya secara otomatis mampu mengelola perubahan data secara efisien melalui `DiffUtil`.

Konstruktor dari `GunungAdapter` menerima tiga parameter fungsi lambda:

- `onLinkClick`: dipanggil saat tombol link ditekan (biasanya membuka URL ke browser).
- `onDetailClick`: dipanggil saat tombol detail ditekan (navigasi ke halaman detail gunung).
- `onFavoriteClick`: dipanggil saat tombol favorit ditekan (menandai/meniadakan gunung sebagai favorit).

Di dalam kelas `GunungAdapter`, terdapat inner class `ListAdapter` yang bertugas mengatur tampilan satu item gunung menggunakan `ItemGunungBinding`, yaitu binding otomatis dari layout XML `item_gunung.xml`. Fungsi `bind()` menerima sebuah objek `Gunung` dan mengatur tampilan setiap elemen UI berdasarkan data tersebut:

- Nama, lokasi, dan deskripsi gunung diatur ke `TextView`.
- Gambar gunung dimuat menggunakan library **Coil** (`binding.imgGunung.load(...)`), yang mengunduh gambar dari URL secara efisien.
- Ikon favorit ditentukan berdasarkan properti `isFavorite`. Jika bernilai `true`, maka ikon bintang penuh ditampilkan, jika `false`, ikon bintang kosong ditampilkan.
- Tiga tombol memiliki listener masing-masing yang memicu fungsi callback dari konstruktor.

Metode `onCreateViewHolder` digunakan untuk *inflate* layout `item_gunung.xml` menjadi objek `ViewHolder`. Sementara itu, `onBindViewHolder` digunakan untuk mengikat data gunung pada posisi tertentu ke holder yang sesuai. Di bagian companion object, terdapat `DIFF_CALLBACK`, yang merupakan implementasi dari `DiffUtil.ItemCallback<Gunung>`. Ini memungkinkan adapter membandingkan item secara otomatis dan hanya memperbarui tampilan yang berubah, bukan seluruh daftar.

- Fungsi `areItemsTheSame()` membandingkan identitas objek berdasarkan nama gunung (yang diasumsikan unik).
- Fungsi `areContentsTheSame()` membandingkan seluruh isi objek (menggunakan `==`, berarti semua properti sama persis).

Dengan pendekatan ini, GunungAdapter memungkinkan RecyclerView untuk tampil optimal, efisien, dan responsif terhadap perubahan data, sekaligus tetap menjaga pemisahan logika tampilan dan logika interaksi. Penggunaan ViewBinding juga mengurangi kemungkinan error dan membuat kode lebih bersih.

13. GunungDetailFragment

Kode di atas merupakan implementasi dari GunungDetailFragment, yaitu sebuah fragment dalam aplikasi Android yang digunakan untuk menampilkan informasi detail mengenai sebuah gunung. Fragment ini menggunakan ViewBinding melalui kelas DetailFragmentBinding untuk menghubungkan elemen-elemen UI di layout XML dengan logika program. Pada metode `onCreateView`, layout di-*inflate* dan disimpan ke dalam variabel `_binding`, yang kemudian diakses melalui properti binding. Dalam proses ini juga dilakukan pengambilan data dari argument yang dikirim oleh fragment sebelumnya, yaitu nama, lokasi, deskripsi, dan URL gambar dari gunung yang dipilih. Data ini diambil dengan memanggil `arguments?.getString()` untuk setiap key yang relevan, disertai nilai default jika datanya tidak tersedia agar tampilan tetap informatif. Setelah data berhasil diambil, isi dari variabel tersebut langsung ditampilkan ke elemen UI yang sesuai.

Nama, lokasi, dan deskripsi ditampilkan pada `TextView`, sementara gambar dimuat ke dalam `ImageView` menggunakan library Coil. Coil memungkinkan pemuatan gambar dari URL dengan fitur transisi lembut menggunakan crossfade, serta mendukung penanganan kondisi saat gambar sedang dimuat (placeholder) maupun saat terjadi kegagalan memuat (error). Selain itu, fragment ini juga memiliki tombol kembali (`btnBack`) yang ketika ditekan akan memanggil `parentFragmentManager.popBackStack()` untuk kembali ke tampilan sebelumnya, serta menyembunyikan kontainer fragment `fragmentContainerGunung` agar tidak tetap tampil di latar belakang. Fragment ini dirancang untuk memberikan pengalaman pengguna yang lebih informatif dan interaktif, terutama ketika pengguna ingin melihat detail

dari sebuah item gunung yang sebelumnya hanya ditampilkan dalam bentuk daftar. Dengan pendekatan berbasis argument dan binding ini, kode menjadi lebih terstruktur, mudah dibaca, dan lebih aman dari kesalahan umum seperti NullPointerException. Fragment ini juga menunjukkan penggunaan prinsip navigasi fragment yang baik di Android, di mana komunikasi antar-fragment dilakukan dengan cara yang eksplisit dan terkontrol.

14. GunungListFragment

Kode di atas merupakan implementasi dari GunungListFragment, yaitu fragment utama yang menampilkan daftar gunung dalam aplikasi Android berbasis arsitektur MVVM (Model-View-ViewModel). Fragment ini menggunakan ViewBinding melalui ListFragmentBinding untuk memudahkan manipulasi elemen UI tanpa perlu menggunakan findViewById. Pada siklus hidup onCreate, fragment menginisialisasi GunungViewModel melalui ViewModelProvider, dengan menyuplai instance GunungRepository dan GunungViewModelFactory. Repository ini menjadi perantara antara sumber data (seperti database atau API) dan ViewModel yang mengelola data gunung. Selanjutnya, pada onViewCreated, fragment menginisialisasi GunungAdapter, yaitu adapter untuk RecyclerView yang menampilkan daftar item gunung. Adapter ini dikonfigurasi dengan tiga aksi interaktif: membuka tautan gunung di browser eksternal saat tombol link ditekan, memicu tampilan detail ketika tombol detail ditekan, dan mengubah status favorit saat ikon bintang ditekan.

Ketika status favorit diubah, objek Gunung disalin menggunakan fungsi copy dan nilainya dibalik, kemudian disalurkan kembali ke ViewModel agar data dapat diperbarui secara reaktif. RecyclerView disusun menggunakan LinearLayoutManager agar daftar ditampilkan secara vertikal. Untuk mengamati data secara real-time, digunakan lifecycleScope.launchWhenStarted yang mengamati tiga aliran data (StateFlow) dari ViewModel: pertama, aliran gunungList untuk memperbarui daftar gunung di adapter; kedua, aliran errorMessage untuk menampilkan pesan kesalahan melalui Toast; dan ketiga, aliran selectedGunung yang merespons pemilihan gunung untuk ditampilkan dalam GunungDetailFragment. Ketika sebuah item gunung dipilih, fragment detail dibuat dan dikirimkan data melalui Bundle sebagai argument, yang kemudian ditransaksikan ke dalam fragmentContainerGunung menggunakan FragmentManager.

Kontainer fragment tersebut juga diatur agar terlihat (VISIBLE) agar tampilan detail dapat ditampilkan secara overlay atau berdampingan dengan daftar. Akhirnya, dalam onDestroyView, binding dihapus untuk mencegah memory leak. Secara keseluruhan, fragment ini mencerminkan pola desain yang bersih dan terstruktur dalam pengembangan aplikasi Android modern. Ia memanfaatkan ViewModel untuk pemisahan logika bisnis dan UI, Coroutine untuk observasi asinkron yang aman terhadap lifecycle, serta fragment dinamis untuk navigasi yang fleksibel antar-tampilan.

15. ViewPagerAdapter

Kode di atas merupakan implementasi dari kelas ViewPagerAdapter, yang digunakan untuk mengelola tampilan halaman (page) pada komponen ViewPager2 di aplikasi Android. Adapter ini merupakan turunan dari FragmentStateAdapter, dan menerima FragmentActivity sebagai parameter utama untuk mengelola siklus hidup fragment yang ditampilkan dalam setiap halaman. Di dalam kelas ini, terdapat dua daftar utama: fragments dan fragmentTitles. Daftar fragments menyimpan dua fragment berbeda, yaitu GunungListFragment yang berfungsi untuk menampilkan daftar semua gunung, serta FavoritGunungFragment yang menampilkan daftar gunung yang telah ditandai sebagai favorit oleh pengguna. Sementara itu, fragmentTitles menyimpan judul masing-masing halaman sesuai urutan fragment, yang nantinya dapat digunakan untuk menampilkan tab dengan label yang sesuai. Metode getItemCount() mengembalikan jumlah total halaman atau fragment yang tersedia, yaitu dua dalam kasus ini.

Sedangkan createFragment(position: Int) bertugas mengembalikan instance fragment berdasarkan posisi yang dipilih oleh pengguna di tampilan tab atau swipe. Selain itu, terdapat fungsi tambahan getPageTitle(position: Int) yang mengembalikan string judul halaman berdasarkan indeksnya. Fungsi ini bisa sangat berguna jika adapter dikombinasikan dengan TabLayout yang menampilkan nama tab berdasarkan posisi. Secara keseluruhan, ViewPagerAdapter ini memberikan cara yang bersih dan terstruktur untuk menyusun tampilan dua halaman dengan ViewPager2, memisahkan antara daftar gunung umum dan daftar favorit, sekaligus memudahkan navigasi pengguna melalui tab atau geser halaman. Adapter ini mendukung modularitas dan pengalaman pengguna yang lebih baik dalam menjelajahi konten yang berbeda namun berkaitan di dalam satu layar.

com/example/modul5/viewmodel

16. GunungViewModel

Kode di atas merupakan implementasi dari kelas GunungViewModel, yang berperan sebagai lapisan logika bisnis dan penghubung antara UI dan data dalam arsitektur MVVM (Model-View-ViewModel) pada aplikasi Android. ViewModel ini bertugas mengelola dan menyimpan data yang berkaitan dengan daftar gunung dan status favoritnya, sekaligus menangani aksi-aksi pengguna seperti memilih gunung atau menandai gunung sebagai favorit. ViewModel ini menerima parameter berupa GunungRepository, yang bertanggung jawab terhadap interaksi data baik dari sumber lokal maupun jarak jauh. Di dalamnya terdapat beberapa StateFlow, yaitu struktur data reaktif yang memungkinkan UI mengamati perubahan secara efisien. `_gunungList` merupakan `MutableStateFlow` privat yang menyimpan daftar semua gunung, lalu dipublikasikan sebagai `gunungList` untuk dapat diamati dari luar. `favoriteList` adalah `StateFlow` yang langsung mengambil data dari repository, menampilkan hanya gunung-gunung yang ditandai sebagai favorit, dan dikelola menggunakan `stateIn()` agar tetap aktif selama ada subscriber.

Selain itu, `_selectedGunung` menyimpan objek gunung yang sedang dipilih oleh pengguna, sementara `_errorMessage` menyimpan pesan kesalahan jika terjadi masalah saat pengambilan atau pemrosesan data. Ketika ViewModel diinisialisasi, fungsi `fetchGunungList()` langsung dijalankan untuk mengambil data gunung dari repository. Fungsi ini dijalankan dalam `viewModelScope` agar aman terhadap siklus hidup dan tidak menyebabkan kebocoran memori. Jika pengambilan data berhasil, maka daftar gunung disimpan dalam `_gunungList`. Jika gagal, pesan kesalahan akan ditampilkan melalui `_errorMessage`. Fungsi `selectGunung()` digunakan saat pengguna memilih salah satu gunung, yang kemudian disimpan ke dalam `selectedGunung`, dan dicatat ke log untuk keperluan debugging. Fungsi `updateGunung()` berfungsi untuk menambah atau menghapus gunung dari daftar favorit berdasarkan statusnya. Setelah melakukan operasi penyimpanan ke repository, daftar gunung juga diperbarui agar perubahan langsung tercermin di UI.

Bila terjadi kesalahan saat menyimpan data, maka akan ditangkap dan dilaporkan ke `errorMessage`. Terakhir, fungsi `onLinkClicked()` digunakan untuk mencatat ke log saat pengguna menekan tautan informasi gunung, meskipun fungsinya bersifat pelengkap dan

tidak memengaruhi data secara langsung. Secara keseluruhan, GunungViewModel menyediakan alur data yang bersih dan terstruktur, memungkinkan UI merespons perubahan secara reaktif. ViewModel ini juga mengedepankan manajemen data yang aman terhadap siklus hidup dengan memanfaatkan coroutine dan StateFlow, serta mendukung pengalaman pengguna yang mulus saat menjelajahi, memilih, atau menandai gunung favorit.

17. GunungViewModelFactory

Kode di atas merupakan implementasi dari kelas GunungViewModelFactory, yang berperan sebagai *factory class* dalam arsitektur MVVM untuk menghasilkan instance dari GunungViewModel. Kelas ini mengimplementasikan interface ViewModelProvider.Factory, yang merupakan standar dalam Jetpack Architecture Components untuk menciptakan ViewModel dengan parameter konstruktor khusus, dalam hal ini sebuah instance GunungRepository. Secara default, ViewModelProvider hanya mampu membuat ViewModel yang memiliki konstruktor kosong. Namun, GunungViewModel membutuhkan GunungRepository sebagai parameter agar dapat mengakses dan mengelola data. Oleh karena itu, diperlukan kelas factory khusus seperti ini untuk menangani pembuatan ViewModel dengan dependensi tersebut. Pada fungsi create, dilakukan pengecekan apakah modelClass yang diminta merupakan turunan dari GunungViewModel.

Jika ya, maka instance GunungViewModel akan dibuat dengan menyisipkan repository sebagai dependensinya, dan hasilnya dikembalikan sebagai tipe T. Supresi terhadap peringatan UNCHECKED_CAST dilakukan karena proses casting secara eksplisit dari GunungViewModel ke tipe generik T. Jika kelas ViewModel yang diminta bukan GunungViewModel, maka factory ini akan melemparkan IllegalArgumentException dengan pesan "Unknown ViewModel class" sebagai penanganan terhadap kemungkinan kesalahan penggunaan factory oleh komponen lain. Dengan pola ini, kode menjadi lebih fleksibel, terstruktur, dan mudah untuk diuji karena dependensi disuntikkan secara eksplisit dan terkontrol. Factory ini juga memungkinkan penerapan prinsip *dependency injection* dengan lebih rapi, dan menjadi bagian penting dalam menjembatani antara lapisan UI (seperti Fragment) dan ViewModel dalam aplikasi.

com/example/modul5/MainActivity.kt

18. MainActivity

Kode di atas merupakan implementasi dari MainActivity, yaitu aktivitas utama dalam aplikasi Android yang menggunakan arsitektur berbasis fragment dan ViewPager2 untuk menampilkan dua halaman berbeda: daftar gunung dan gunung favorit. Aktivitas ini memanfaatkan ActivityMainBinding, yang merupakan bagian dari View Binding, untuk mengakses elemen-elemen tampilan yang didefinisikan dalam layout activity_main.xml secara aman dan efisien tanpa perlu menggunakan findViewById() secara manual. Di dalam metode onCreate, dilakukan inisialisasi binding dengan memanggil inflate() pada ActivityMainBinding, yang kemudian dijadikan tampilan utama melalui setContentView(binding.root). Selanjutnya, dibuat instance dari ViewPagerAdapter, yaitu adaptor khusus yang menangani logika fragment pada ViewPager2. Adapter ini kemudian dihubungkan dengan binding.viewPager sehingga pengguna dapat menggeser antar halaman secara horizontal. Agar ViewPager2 dapat berfungsi dengan baik bersama TabLayout, digunakan TabLayoutMediator.

Mediator ini menjembatani antara tabLayout dan viewPager dengan menetapkan judul tab berdasarkan posisi, yang diambil dari metode getPageTitle(position) dalam ViewPagerAdapter. Fungsi attach() di akhir baris berfungsi untuk mengikat tab dengan halaman ViewPager secara penuh, sehingga pengguna dapat menavigasi ke fragment yang berbeda baik dengan menggeser halaman maupun mengetuk tab yang sesuai. Secara keseluruhan, MainActivity ini bertindak sebagai container utama yang mengatur alur navigasi antar fragment menggunakan tab, ViewPager2, dan View Binding, sehingga UI terasa modern, intuitif, dan mudah dikembangkan lebih lanjut.

MODUL5/app/src/main/res/layout

19. activity_main.xml

Kode XML di atas adalah definisi layout utama untuk MainActivity pada aplikasi Android berbasis fragment, ViewPager2, dan tab navigasi. Layout ini menggunakan ConstraintLayout sebagai root view, yang memungkinkan penempatan komponen UI secara

fleksibel dengan constraint antar elemen, sehingga tampilan tetap responsif di berbagai ukuran layar. Di bagian atas layout, terdapat elemen TabLayout dari Material Components yang berfungsi sebagai indikator tab navigasi. Elemen ini memiliki lebar yang menyesuaikan parent (0dp dengan constraint start dan end) dan tinggi yang disesuaikan dengan kontennya. Beberapa atribut styling digunakan untuk mengatur warna indikator tab, teks tab saat dipilih (tabSelectedTextColor), dan teks tab saat tidak dipilih (tabTextColor), di mana warna-warna ini diambil dari resource seperti @color/blue_500 dan @color/black.

Di bawah TabLayout, terdapat ViewPager2, yang merupakan komponen untuk menampilkan fragment secara horizontal melalui geseran (swipe). View ini dihubungkan dengan TabLayout melalui TabLayoutMediator di kelas MainActivity, sehingga setiap tab merepresentasikan fragment tertentu. ViewPager2 ini juga dikonstrain ke semua sisi parent, kecuali bagian atas yang dikaitkan ke TabLayout. Komponen ketiga adalah FrameLayout dengan ID fragmentContainerGunung. View ini disiapkan sebagai container tambahan untuk menampilkan fragment detail (misalnya GunungDetailFragment) yang muncul saat pengguna memilih item dari daftar gunung. Awalnya, FrameLayout ini disembunyikan dengan android:visibility="gone" dan hanya akan dimunculkan ketika ada aksi navigasi ke detail, yang diatur melalui logika di kelas fragment seperti GunungListFragment.

Secara keseluruhan, layout ini dirancang agar fleksibel dalam mendukung navigasi antara fragment menggunakan tab dan ViewPager2, serta memberikan ruang tambahan untuk menampilkan fragment detail dengan cara overlay yang dapat dikontrol visibilitasnya. Pendekatan ini membuat UI terasa terstruktur, terorganisir, dan siap untuk pengembangan aplikasi yang modular dan interaktif.

20. detail_fragment.xml

Layout ini merupakan tampilan detail dari sebuah gunung yang ditampilkan menggunakan ScrollView, memungkinkan pengguna menggulir konten secara vertikal, terutama jika deskripsi atau tampilan gambar terlalu panjang untuk satu layar. Root layout ScrollView menggunakan atribut match_parent untuk lebar dan tinggi agar memenuhi seluruh ruang layar, serta diberi latar belakang berwarna biru muda (#B6D6F1) yang

memberikan kesan sejuk dan natural, sesuai dengan tema alam dari konten gunung. Padding sebesar 16dp di sekeliling isi layout membantu menciptakan ruang yang nyaman antara tepi layar dan elemen konten, mencegah tampilan menjadi terlalu rapat. Di dalam ScrollView, terdapat sebuah LinearLayout vertikal yang menyusun seluruh elemen UI dari atas ke bawah. Elemen pertama adalah ImageView dengan ID imgPoster, yang menampilkan gambar gunung berukuran tetap setinggi 300dp dan mengisi seluruh lebar layar.

Atribut scaleType="centerCrop" memastikan gambar di-zoom dan dipotong agar memenuhi ukuran tampilan tanpa mengubah rasio aspek, menghasilkan tampilan visual yang estetis. Selanjutnya, terdapat tiga buah TextView untuk menampilkan nama gunung (tvName), lokasi (tvLokasi), dan deskripsi (tvDeskripsi). Masing-masing teks memiliki penyesuaian tampilan seperti ukuran font besar pada nama (35sp dengan bold) untuk menonjolkan identitas utama gunung, serta sentuhan gravity="center" untuk menyeimbangkan teks secara horizontal pada layar. Untuk teks lokasi, digunakan ukuran sedang (22sp) dan margin atas untuk memberikan jeda antar komponen, sedangkan deskripsi gunung menggunakan justificationMode="inter_word" (berlaku mulai API 26 ke atas) yang meratakan teks secara horizontal seperti pada paragraf di koran atau buku, membuat deskripsi panjang tampak lebih rapi dan mudah dibaca. Terakhir, di bagian bawah terdapat Button dengan ID btnBack yang berfungsi sebagai kontrol navigasi untuk kembali ke tampilan sebelumnya. Button ini diposisikan di tengah menggunakan layout_gravity="center" dan diberi margin atas agar tidak terlalu dekat dengan teks deskripsi. Secara keseluruhan, layout ini dirancang dengan keseimbangan estetika dan fungsionalitas.

Setiap elemen ditata secara rapi untuk menampilkan informasi penting mengenai gunung secara visual dan tekstual, serta disiapkan untuk berjalan mulus di berbagai ukuran layar dengan bantuan elemen scroll dan layout yang responsif. Kombinasi warna, ukuran font, margin, dan tata letak membuat tampilan ini informatif namun tetap menarik secara visual.

21. fragment_favorit_gunung.xml

File XML ini mendefinisikan tampilan antarmuka pengguna berbasis ConstraintLayout, yang merupakan salah satu layout paling fleksibel dan efisien di Android karena memungkinkan penempatan komponen UI secara relatif terhadap satu sama lain

maupun terhadap parent-nya. Layout ini memiliki lebar dan tinggi yang disetel ke match_parent, yang berarti akan memenuhi seluruh ruang layar perangkat, memastikan semua konten tertata secara penuh tanpa batasan ruang. Di dalam ConstraintLayout hanya terdapat satu elemen utama yaitu RecyclerView dengan ID recyclerViewFavorit, yang berfungsi untuk menampilkan daftar item favorit dalam bentuk list atau grid yang bisa digulir secara vertikal atau horizontal. Penggunaan RecyclerView di sini menandakan bahwa daftar yang ditampilkan bersifat dinamis dan dapat memuat banyak data secara efisien, karena RecyclerView hanya membuat tampilan sebanyak yang diperlukan di layar, sementara sisanya didaur ulang (recycled), menghemat memori dan meningkatkan performa. Lebar dan tinggi RecyclerView diatur dengan 0dp (alias match constraints), artinya ukurannya akan disesuaikan berdasarkan constraint yang diberikan. Empat atribut app:layout_constraintTop_toTopOf, app:layout_constraintBottom_toBottomOf, app:layout_constraintStart_toStartOf, dan app:layout_constraintEnd_toEndOf semuanya mengacu ke parent, yaitu ConstraintLayout itu sendiri.

Dengan pengaturan ini, RecyclerView akan menempati seluruh ruang dari atas ke bawah dan dari kiri ke kanan pada layar, menjadikannya komponen yang dominan dalam tampilan ini. Secara visual, layout ini sangat minimalis dan efisien, hanya berisi elemen tunggal yang bertugas menampilkan daftar favorit pengguna. Struktur seperti ini umum digunakan dalam fragment atau halaman khusus dalam aplikasi Android yang bertugas menyajikan data dalam format list, seperti daftar gunung favorit, item yang disimpan, atau konten yang sering diakses pengguna. Kode ini juga fleksibel untuk dikembangkan lebih lanjut, seperti menambahkan elemen pendukung (misalnya Toolbar, SearchView, atau FloatingActionButton) jika diinginkan dalam pengembangan berikutnya.

22. item_gunung.xml

File XML ini mendefinisikan struktur visual untuk satu kartu data menggunakan CardView sebagai kontainer utama. CardView digunakan karena mampu memberikan efek bayangan (cardElevation) dan sudut membulat (cardCornerRadius), menciptakan tampilan yang modern dan rapi, serta meningkatkan keterbacaan elemen-elemen di dalamnya. Kartu ini dirancang untuk menampilkan informasi singkat mengenai gunung, meliputi gambar,

nama, lokasi, deskripsi pendek, dan tiga tombol aksi: link, detail, serta tombol favorit berbentuk ikon. Di dalam CardView, terdapat sebuah ConstraintLayout yang digunakan sebagai layout utama karena fleksibilitasnya dalam menempatkan elemen-elemen UI secara presisi. ConstraintLayout ini memiliki padding untuk memberikan ruang antara konten dan tepi kartu, serta latar belakang berwarna krem (@color/cream) untuk mempertegas batas visual antar item. Di sisi kiri terdapat ImageView berukuran 120x160dp untuk menampilkan foto gunung, dengan pengaturan centerCrop agar gambar terfokus pada bagian tengah. Sebelah kanan gambar terdapat tiga TextView berisi nama gunung (tvGunungName) dengan ukuran font besar dan tebal untuk penekanan, lokasi (tvGunungLokasi) yang ditampilkan lebih kecil, dan deskripsi (tvGunungDeskripsi) yang dibatasi maksimal dua baris dan akan terpotong dengan ellipsis jika terlalu panjang, menjaga tampilan tetap ringkas.

Ketiganya disejajarkan secara vertikal dan dikaitkan dengan gambar menggunakan constraint start/end, menciptakan keselarasan visual yang konsisten. Selanjutnya, terdapat tiga komponen aksi. Tombol btnLink memungkinkan pengguna membuka tautan eksternal atau navigasi tambahan terkait gunung tersebut, btnDetail untuk melihat detail lengkap, dan btnFavorite berupa ImageButton berbentuk bintang untuk menandai gunung sebagai favorit. Tombol-tombol ini diposisikan sejajar di bawah deskripsi menggunakan constraint, dan bahkan btnFavorite diikat ke sisi kanan parent untuk menempati ujung baris. Tombol-tombol tersebut memiliki warna yang kontras seperti @color/teal_700, @color/blue, dan @color/pink, serta ukuran dan padding yang konsisten untuk menjaga keteraturan dan aksesibilitas. Komponen tambahan berupa Barrier (barrierStartText) digunakan untuk menyelaraskan posisi horizontal tombol berdasarkan ujung kiri teks, memberikan fleksibilitas dalam responsivitas tampilan, terutama jika ada perubahan panjang teks. Elemen ini merupakan teknik lanjutan di ConstraintLayout untuk menjaga konsistensi antar elemen meski konten bersifat dinamis.

Secara keseluruhan, layout ini dirancang tidak hanya untuk tampil menarik, tetapi juga responsif, mudah dinavigasi, dan mendukung pengalaman pengguna yang baik. Kombinasi antara gambar, teks, dan tombol aksi menjadikan setiap kartu informatif sekaligus interaktif, cocok untuk aplikasi katalog wisata alam atau aplikasi data gunung.

23. list_fragment.xml

Layout ini mendefinisikan tampilan utama dari sebuah fragmen (GunungListFragment) dalam aplikasi Android berbasis Jetpack, yang menggunakan ConstraintLayout sebagai root-nya. ConstraintLayout dipilih karena mampu menyusun komponen UI secara fleksibel dan efisien dengan menggunakan constraint antar elemen atau terhadap parent. Layout ini memiliki lebar dan tinggi yang mengisi seluruh layar (match_parent) dan disertai atribut tools:context untuk memberikan referensi konteks saat preview di Android Studio, yaitu GunungListFragment. Di dalamnya terdapat sebuah RecyclerView dengan ID rvGunung, yang berfungsi sebagai komponen utama untuk menampilkan daftar gunung secara vertikal dalam bentuk scrollable list. RecyclerView ini diatur agar lebarnya mengisi penuh dari kiri ke kanan (0dp dengan constraint start/end) dan tingginya dari atas ke bawah (0dp dengan constraint top/bottom), menempel langsung ke batas atas, bawah, kiri, dan kanan ConstraintLayout. RecyclerView ini juga diberi padding sebesar 16dp di semua sisi untuk menciptakan ruang antara isi dan batas layout, serta clipToPadding="false" agar konten seperti efek bayangan (elevation) dari item tetap terlihat di area padding tersebut.

Warna latar belakang RecyclerView diatur ke warna biru muda (#94C2EB), memberikan nuansa visual yang cerah dan mendukung pengalaman pengguna yang menyenangkan. Untuk keperluan preview dan pengembangan di Android Studio, atribut tools:listItem="@layout/item_gunung" ditambahkan. Ini memungkinkan pengembang melihat tampilan representatif dari item yang akan ditampilkan di daftar, berdasarkan layout item_gunung.xml. Secara keseluruhan, layout ini memfasilitasi sebuah daftar gunung dengan desain yang responsif, bersih, dan estetis, siap diisi dengan data dinamis menggunakan adapter di dalam fragment terkait. Struktur ini merupakan fondasi dari user interface berbasis daftar, yang umum digunakan dalam aplikasi pencarian, katalog, atau eksplorasi destinasi wisata alam.

res/values

24. colors.xml

Kelas XML ini berfungsi sebagai definisi sumber daya warna (colors.xml) dalam proyek Android. File ini terletak di direktori res/values/ dan digunakan untuk menyimpan nilai warna yang dapat dipanggil di seluruh aplikasi menggunakan @color/nama_warna. Dalam file ini, terdapat sejumlah entri warna yang diberi nama tertentu agar mudah digunakan dan dikelola secara konsisten. Misalnya, blue_500 memiliki nilai warna #2196F3, yang merupakan salah satu nuansa biru cerah dari palet material design. Warna blue_700, black, blue, pink, teal_700, dan white semuanya diberikan kode warna #1976D2, yaitu warna biru tua yang sama, meskipun namanya berbeda.

Hal ini menunjukkan bahwa proyek ini kemungkinan masih dalam tahap pengembangan atau eksperimen warna, di mana nama-nama placeholder seperti black, white, atau pink masih belum disesuaikan dengan makna warna sebenarnya. Satu-satunya warna yang berbeda adalah cream, yang menggunakan nilai #FFEBCD, yakni warna krem muda yang lembut dan netral, sangat cocok sebagai latar belakang untuk meningkatkan keterbacaan teks dan memberi nuansa hangat pada desain UI. Pendefinisian warna dalam file ini memungkinkan penggunaan warna yang konsisten dan efisien di berbagai elemen layout, style, maupun komponen UI lainnya dalam aplikasi Android. Namun, agar tidak membingungkan, sebaiknya nama warna disesuaikan dengan nilai warnanya untuk menjaga keterbacaan dan maintainability kode.

25. string.xml

File strings.xml merupakan bagian dari sumber daya (resources) dalam proyek Android yang digunakan untuk menyimpan teks statis seperti label, judul, dan deskripsi agar mudah diatur dan diterjemahkan jika aplikasi nantinya mendukung berbagai bahasa (internationalization). Dalam file ini terdapat beberapa string penting. app_name berisi nilai "MODUL 5", yang menandakan nama aplikasi atau proyek yang sedang dikembangkan. Kemudian, gunung_image_desc berisi deskripsi alternatif "Gambar gunung", yang biasanya digunakan untuk keperluan aksesibilitas atau konten deskriptif pada elemen gambar (ImageView) agar dapat dikenali oleh screen reader.

Selanjutnya, terdapat tiga string untuk teks tombol: `btn_detail_text` dengan nilai "Detail" yang kemungkinan menampilkan informasi lengkap suatu gunung, `btn_favorite_text` dengan karakter bintang kosong "☆" yang digunakan sebagai ikon tombol favorit, serta `btn_link_text` yang bertuliskan "Link", kemungkinan mengarah ke tautan eksternal atau informasi tambahan. Penggunaan string resource seperti ini sangat penting untuk menjaga keteraturan kode, memudahkan proses perubahan konten teks, serta memungkinkan dukungan multi-bahasa secara efisien di masa depan.

res/values/themes.xml

26. themes.xml

File `themes.xml` ini berfungsi untuk mendefinisikan tema global aplikasi Android kamu, yaitu *MODUL 5*. Di dalamnya terdapat dua style: `Base.Theme.Modul5` dan `Theme.Modul5`. Style `Base.Theme.Modul5` menjadi fondasi utama dari tema aplikasi, yang mewarisi dari `Theme.Material3.DayNight.NoActionBar`. Artinya, aplikasi ini mengadopsi gaya Material Design 3 (Material You) dengan dukungan mode gelap terang otomatis, serta tidak menggunakan *ActionBar* bawaan. Tema ini juga mengatur beberapa elemen warna utama, seperti `colorPrimary` yang mengambil warna dari `@color/blue_500`, serta `colorPrimaryContainer` dari `@color/blue_700`, yang biasanya digunakan untuk elemen UI seperti tombol dan latar belakang container. Selain itu, `colorOnPrimary` dan `colorOnPrimaryContainer` diatur ke warna putih (`@android:color/white`) agar teks atau ikon yang berada di atas elemen dengan latar belakang biru tetap terlihat jelas dan kontras.

Kemudian, style `Theme.Modul5` dideklarasikan sebagai turunan langsung dari `Base.Theme.Modul5`, yang artinya kamu bisa menggunakan其 sebagai tema utama aplikasi di `AndroidManifest.xml`. Struktur ini memberi fleksibilitas jika suatu saat kamu ingin membuat variasi tema lainnya tanpa harus menulis ulang seluruh konfigurasi warna atau atribut tampilan. Singkatnya, file ini menetapkan identitas visual aplikasi agar tampil konsisten dengan sentuhan warna biru khas yang sudah kamu definisikan di file `colors.xml`.

MODUL5/app/build.gradle.kts

27. build.gradle.kts

File ini adalah konfigurasi utama proyek Android kamu dalam format Kotlin DSL, yang mendefinisikan bagaimana proyek dikompilasi, dependensi yang digunakan, serta fitur yang diaktifkan. Di bagian plugins, kamu mengaktifkan beberapa plugin penting seperti com.android.application untuk aplikasi Android, org.jetbrains.kotlin.android untuk dukungan Kotlin di Android, kotlin-parcelize untuk kemudahan pengiriman objek melalui Intent, kotlin-kapt untuk annotation processing (khususnya digunakan oleh Room), serta kotlin.plugin.serialization untuk serialisasi/deserialisasi data menggunakan KotlinX Serialization. Di blok android, namespace proyek ditentukan sebagai com.example.modul5, dengan compileSdk 34 dan targetSdk serta minSdk masing-masing 34 dan 30, yang memastikan aplikasi berjalan di Android 11 ke atas. Build type release menonaktifkan ProGuard minification untuk memudahkan debug. Bagian compileOptions dan kotlinOptions menunjukkan bahwa proyek ini menggunakan Java 17 dan Kotlin dengan target JVM 17, yang berarti fitur-fitur modern Java/Kotlin bisa digunakan. viewBinding juga diaktifkan, memungkinkan binding langsung ke view dari layout XML tanpa menggunakan findViewById.

Pada blok dependencies, proyek menggunakan banyak library modern. Untuk core Android, digunakan core-ktx, appcompat, fragment-ktx, dan material untuk Material Design. Untuk UI, digunakan constraintlayout, recyclerview, dan cardview. Manajemen lifecycle difasilitasi oleh lifecycle-runtime-ktx,.viewmodel-ktx, dan livedata-ktx. Proyek juga menggunakan Retrofit untuk networking (retrofit, kotlinx-serialization-json, dan retrofit2-kotlinx-serialization-converter) serta okhttp3-logging-interceptor untuk memudahkan debugging HTTP. Untuk memuat gambar, digunakan Coil versi 2.5.0. Untuk penyimpanan data lokal, proyek menggunakan Room database (room-runtime, room-compiler via kapt, dan room-ktx), serta mendukung coroutine lewat kotlinx-coroutines. Proyek juga menggunakan datastore-preferences sebagai cara modern menggantikan SharedPreferences. Terakhir, tersedia dependensi untuk testing, yaitu JUnit untuk unit test serta androidx.test dan espresso untuk pengujian UI. Konfigurasi ini menunjukkan bahwa proyek kamu sudah cukup

lengkap dan modern, mendukung arsitektur berbasis MVVM, konektivitas API, manajemen database lokal, dan penanganan gambar secara efisien.

TAUTAN GIT

<https://github.com/SheilaSabina/Praktikum-Mobile>