

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3**



BUILD A SCROLLABLE LIST

Oleh:

Sheila Sabina

NIM. 2310817220028

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

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 3

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Sheila Sabina
NIM : 2310817220028

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Zulfa Auliya Akbar
NIM. 2210817210026

Muti`a Maulida S.Kom M.T.I
NIP. 19881027 201903 20 13

DAFTAR ISI

LEMBAR PENGESAHAN	2
DAFTAR ISI	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL	5
SOAL 1	6
A. Source Code.....	8
B. Output Program	23
C. Pembahasan	25
D. Tautan Git.....	44

DAFTAR GAMBAR

Gambar 1. Screenshot Hasil Jawaban Soal 1	25
---	----

DAFTAR TABEL

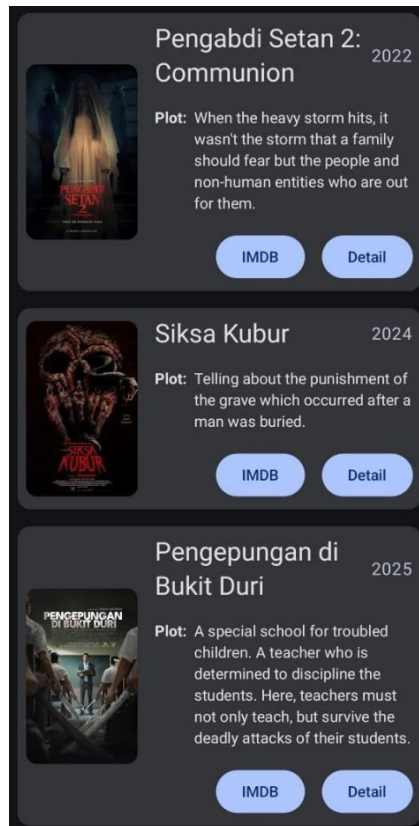
Tabel 1. Source Code DetailFragment.kt	8
Tabel 2. Source Code Gunung.kt.....	9
Tabel 3. Source Code GunungAdapter.kt.....	10
Tabel 4. Source Code ListFragment.kt.....	12
Tabel 5. Source Code MainActivity.kt.....	12
Tabel 6. Source Code activity_main.xml	13
Tabel 7. Source Code detail_fragment.xml	14
Tabel 8. Source Code item_gunung.xml	16
Tabel 9. Source Code list_fragment.xml	17
Tabel 10. Source Code colors.xml.....	17
Tabel 11. Source Code string.xml	19
Tabel 12. Source Code theme.xml.....	20
Tabel 13. Source Code AndroidManifest.xml.....	21
Tabel 14. Source Code BuildGradle.kts	22

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

```
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 = DetailFragmentBinding.inflate(inflater,
20 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 = arguments?.getString("EXTRA_DESKRIPSI")
26
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
35         return binding.root
36     }
37
38     override fun onDestroyView() {
39         super.onDestroyView()
40         _binding = null
41     }
42 }
```

Tabel 1. Source Code DetailFragment.kt

2. 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
```

Tabel 2. Source Code Gunung.kt

3. 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 }

```

Tabel 3. Source Code GunungAdapter.kt

4. 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             onClick = { 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	

Tabel 4. Source Code ListFragment.kt

5. 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	}

Tabel 5. Source Code MainActivity.kt

app/src/main/res/layout

6. 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>

Tabel 6. Source Code activity_main.xml

7. 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>

Tabel 7. Source Code detail_fragment.xml

8. 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>

Tabel 8. Source Code item_gunung.xml

9. 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	

22	tools:listitem="@layout/item_gunung" />
23	</androidx.constraintlayout.widget.ConstraintLayout>

Tabel 9. Source Code list_fragment.xml

app\src\main\res\values

10. 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 10. Source Code colors.xml

11. 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.rinjaninationalpark.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	<code><item></code> 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. <code></item></code>
108	<code><item></code> 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	<code></item></code>
128	<code></string-array></code>
129	
130	<code><array name="gunung_image"></code>
131	<code><item></code> @drawable/gunung_bromo <code></item></code>
132	<code><item></code> @drawable/gunung_kerinci <code></item></code>
133	<code><item></code> @drawable/gunung_merapi <code></item></code>
134	<code><item></code> @drawable/gunung_merbabu <code></item></code>
135	<code><item></code> @drawable/gunung_rinjani <code></item></code>
136	<code></array></code>
137	<code></resources></code>

Tabel 11. Source Code string.xml

12. 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>
```

Tabel 12. Source Code theme.xml

app\src\main

13. 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 }

```

Tabel 13. Source Code AndroidManifest.xml

14. build.gradle.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(

```

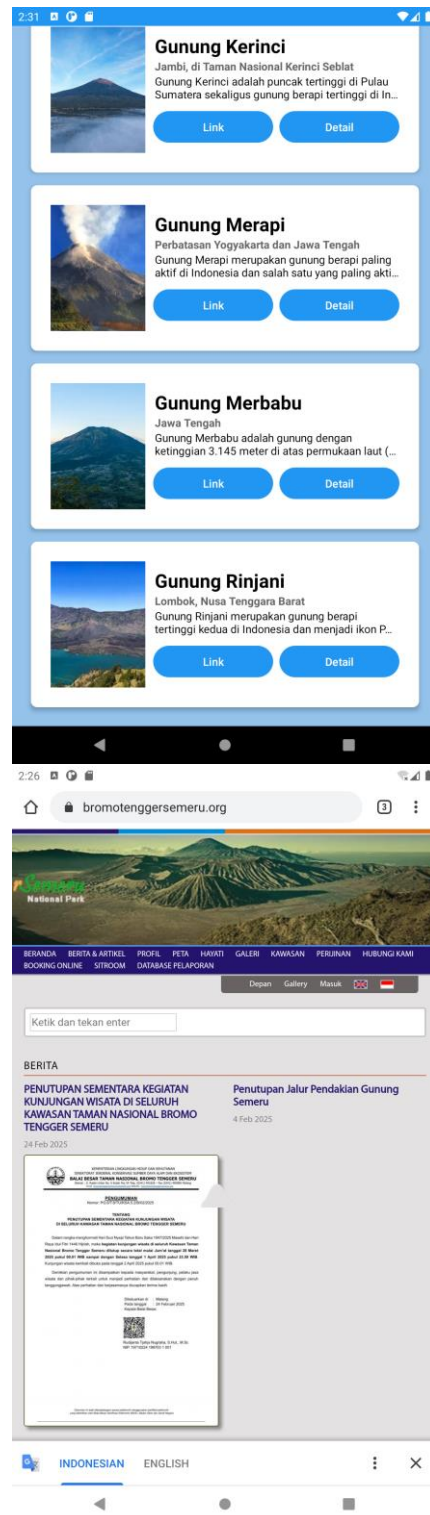
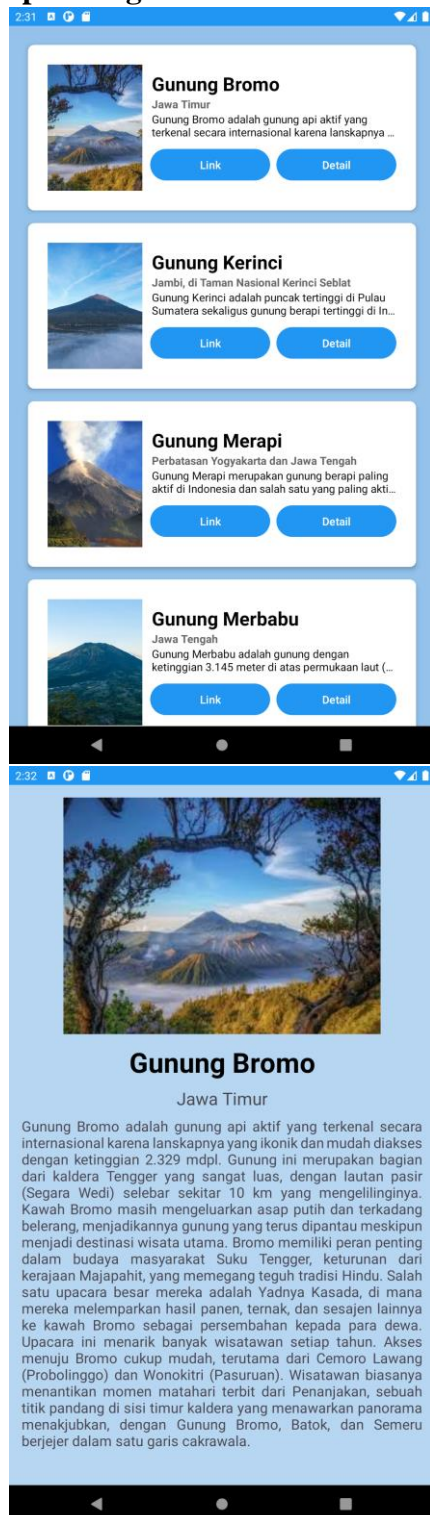
```

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 }


```

Tabel 14. Source Code BuildGradle.kts

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 Nasiona...

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

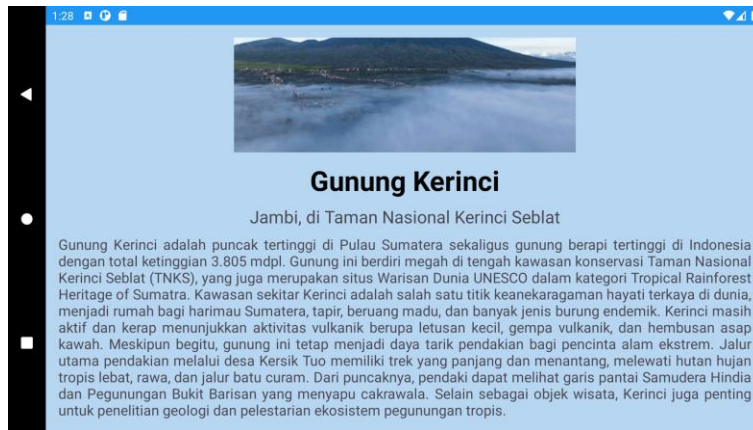
2:35



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 1. Screenshot Hasil Jawaban Soal 1

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 `ViewHolder` 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 `ViewHolder` 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 `ViewHolder`, 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 menggulir. 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 `onDestroyView()` 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 !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`

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 hirarki 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). Lebar nya 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 estetik. Pada baris [39] hingga [49] terdapat fungsi TextView ini digunakan untuk menampilkan lokasi gunung dan ditempatkan tepat di bawah nama gunung. Lebar nya 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. Lebaranya 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. Lebar nya 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 ditambahkan 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 aksent 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 `ViewHolder` sebagai `ViewHolder` untuk mengikat tampilan item. Pada baris [15] hingga [30] fungsi `ViewHolder` 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.btnLink` 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.

14. BuilGradle.kts

Kode kelas ini secara keseluruhan berguna untuk mengatur 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).

D. Tautan Git

<https://github.com/SheilaSabina/Praktikum-Mobile/tree/master/MODUL3>

SOAL 2

Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

Jawab:

RecyclerView masih banyak digunakan dalam pengembangan aplikasi Android karena beberapa alasan yang cukup kuat, meskipun saat ini sudah ada alternatif seperti LazyColumn dari Jetpack Compose yang menawarkan kode lebih singkat dan efisien. Pertama, RecyclerView adalah bagian dari View System (XML-based UI), yang masih menjadi fondasi utama di banyak proyek Android, terutama pada aplikasi-aplikasi lama atau yang sudah dikembangkan jauh sebelum Compose diperkenalkan. Migrasi penuh dari View System ke Compose bukanlah hal yang sederhana, karena melibatkan banyak perubahan arsitektur dan komponen, sehingga banyak tim developer tetap mempertahankan RecyclerView untuk menjaga stabilitas dan kompatibilitas aplikasi mereka. Kedua, meskipun RecyclerView memerlukan sedikit lebih banyak kode (seperti ViewHolder, Adapter, dll.), struktur ini justru memberikan fleksibilitas dan kontrol lebih dalam mengelola tampilan kompleks, misalnya daftar dengan banyak jenis item, animasi kustom, nested scroll, hingga optimalisasi performa untuk data yang sangat besar. RecyclerView juga telah matang secara fitur dan dokumentasi, serta memiliki ekosistem pendukung yang luas. Ketiga, tidak semua tim atau proyek sudah siap menggunakan Jetpack Compose karena faktor seperti waktu, sumber daya, atau kebutuhan untuk tetap menggunakan pendekatan tradisional (misalnya jika desain UI-nya sangat kompleks dan telah diatur dalam XML). Dengan kata lain, RecyclerView tetap menjadi pilihan yang solid dan andal dalam banyak konteks, terutama jika proyek dibangun dengan pendekatan konvensional yang sudah mapan. Jadi meskipun LazyColumn sangat menawarkan kemudahannya, RecyclerView masih relevan karena fleksibilitas, kestabilan, dan kompatibilitas jangka panjang yang ditawarkannya.