LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Allano Lintang Ertantora **NIM. 2310817210004**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
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 : Allano Lintang Ertantora
NIM : 2310817210004

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

DAFTAR GAMBAR

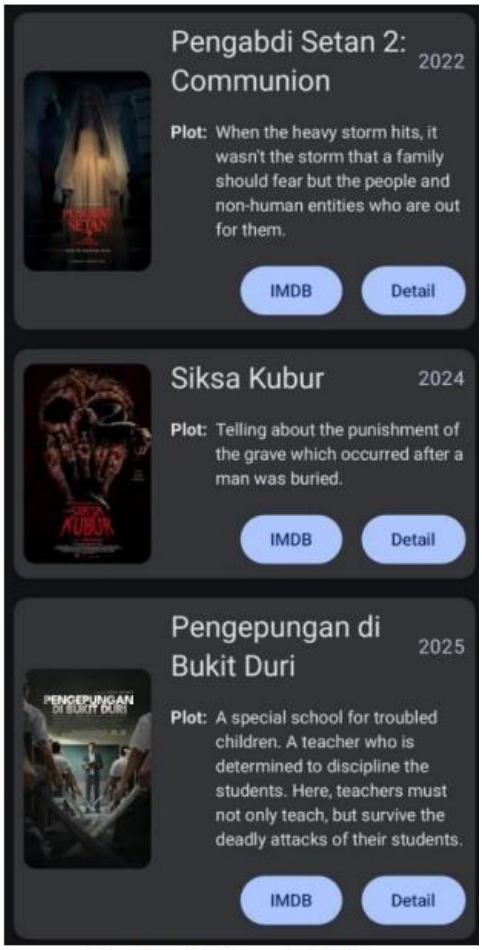
DAFTAR TABEL

SOAL 1

Soal Praktikum:

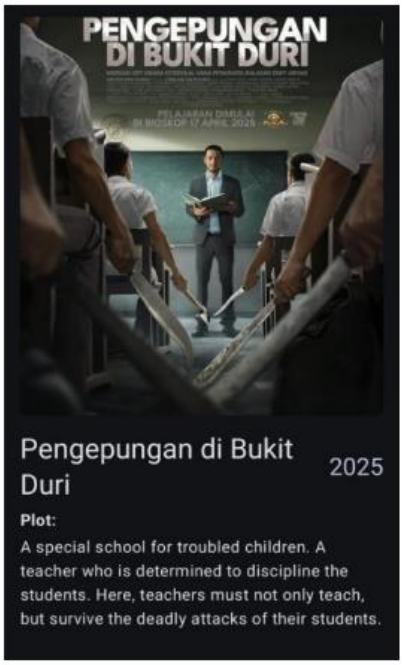
1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3. Item pada 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

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.activity.enableEdgeToEdge
6   import com.allano.nongki.databinding.ActivityMainBinding
7
8   class MainActivity : AppCompatActivity() {
9       private lateinit var binding: ActivityMainBinding
10
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          enableEdgeToEdge()
14          binding =
    ActivityMainBinding.inflate(layoutInflater)
15          setContentView(binding.root)
16      }
17  }
```

2. HomeFragment.kt

Tabel 2. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.os.Bundle
4  import android.view.LayoutInflater
5  import android.view.View
6  import android.view.ViewGroup
7  import androidx.fragment.app.Fragment
8  import androidx.fragment.app.viewModels
9  import androidx.navigation.fragment.findNavController
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import com.allano.nongki.databinding.FragmentHomeBinding
12
13 class HomeFragment : Fragment() {
14     private var _binding: FragmentHomeBinding? = null
15     private val binding get() = _binding!!
16
17     private val viewModel: LocationViewModel by
   viewModels()
18
19     override fun onCreateView(inflater: LayoutInflater,
   container: ViewGroup?, savedInstanceState: Bundle?): View?
   {
20         _binding = FragmentHomeBinding.inflate(inflater,
   container, false)
21         return binding.root
22     }
23
24     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
25         val adapter =
26 LocationAdapter(viewModel.getLocations(),
   findNavController())
27         binding.recyclerView.layoutManager =
   LinearLayoutManager(requireContext())
28         binding.recyclerView.adapter = adapter
29     }
30
31     override fun onDestroyView() {
32         super.onDestroyView()
33         _binding = null
34     }
35
36 }
```

3. LocationViewModel.kt

Tabel 3. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.app.Application
4   import androidx.lifecycle.AndroidViewModel
5
6   class LocationViewModel(application: Application) :
    AndroidViewModel(application){
7       private val context =
    getApplication<Application>().applicationContext
8
9       fun getLocations(): List<LocationModel> {
10          val nama =
    context.resources.getStringArray(R.array.data_name)
11          val deskripsi =
    context.resources.getStringArray(R.array.data_description)
12          val rating =
    context.resources.getStringArray(R.array.data_rating)
13          val link =
    context.resources.getStringArray(R.array.data_link)
14          val fotoResId = listOf(
15              R.drawable.image1,
16              R.drawable.image2,
17              R.drawable.image3,
18              R.drawable.image4,
19              R.drawable.image5
20          )
21
22          return nama.indices.map { i ->
23              LocationModel(
24                  nama[i],
25                  fotoResId[i],
26                  deskripsi[i],
27                  rating[i],
28                  link[i]
29              )
30          }
31      }
32  }
```

4. LocationAdapter.kt

Tabel 4. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.content.Intent
4  import android.view.LayoutInflater
5  import android.view.ViewGroup
6  import androidx.navigation.NavController
7  import androidx.recyclerview.widget.RecyclerView
8  import com.allano.nongki.databinding.ItemLocationBinding
9  import androidx.core.net.toUri
10
11 class LocationAdapter(
12     private val items: List<LocationModel>,
13     private val navController: NavController
14 ) : RecyclerView.Adapter<LocationAdapter.LocationViewHolder>() {
15
16     class LocationViewHolder(val binding: ItemLocationBinding) :
   RecyclerView.ViewHolder(binding.root)
17
18     override fun onCreateViewHolder(parent: ViewGroup, viewType:
   Int): LocationViewHolder {
19         val binding =
   ItemLocationBinding.inflate(LayoutInflater.from(parent.context),
   parent, false)
20         return LocationViewHolder(binding)
21     }
22
23     override fun onBindViewHolder(holder: LocationViewHolder,
   position: Int) {
24         val item = items[position]
25         holder.binding.textViewName.text = item.nama
26         holder.binding.rating.text = item.rating
27
   holder.binding.imageView.setImageResource(item.fotoResId)
28
29         holder.binding.buttonBrowser.setOnClickListener {
30             val url = item.link
31             val intent = Intent(Intent.ACTION_VIEW, url.toUri())
32             it.context.startActivity(intent)
33         }
34
35         holder.binding.buttonDetail.setOnClickListener {
36             val bundle = android.os.Bundle().apply {
37                 putInt("fotoResId", item.fotoResId)
38                 putString("nama", item.nama)
39                 putString("deskripsi", item.deskripsi)
40             }
```

```
41              navController.navigate(R.id.detailFragment, bundle)
42          }
43      }

        override fun getItemCount() = items.size
44  }
45
46
```

5. DetailFragment.kt

Tabel 5. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
5   import android.view.View
6   import android.view.ViewGroup
7   import androidx.fragment.app.Fragment
8   import com.allano.nongki.databinding.FragmentDetailBinding
9
10  class DetailFragment: Fragment() {
11      private var _binding: FragmentDetailBinding? = null
12      private val binding get() = _binding!!
13
14      override fun onCreateView(inflater: LayoutInflater,
    container: ViewGroup?, savedInstanceState: Bundle?):
    View?{
15          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
16          return binding.root
17      }
18
19      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
20          val nama = arguments?.getString("nama") ?: "Tidak
    ada nama"
21          val deskripsi = arguments?.getString("deskripsi")
    ?: "Tidak ada deskripsi"
22          val fotoResId = arguments?.getInt("fotoResId") ?:
    R.drawable.image5
23         binding.imageViewDetail.setImageResource(fotoResId)
24          binding.titleViewDetail.text = nama
25          binding.textViewDetail.text = deskripsi
26      }
27
```

```
    override fun onDestroyView() {
        super.onDestroyView()
        _binding = null
    }
}
```

6. LocationAdapter.kt

Tabel 6. Source Code Jawaban Soal 1

```
package com.allano.nongki

data class LocationModel (
    val nama: String,
    val fotoResId: Int,
    val deskripsi: String,
    val rating: String,
    val link: String
)
```

7. Activity_main.xml

Tabel 7. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    app:navGraph="@navigation/nav_graph"
    app:defaultNavHost="true" />
```

8. Fragment_home.xml

Tabel 8. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```
8       <androidx.recyclerview.widget.RecyclerView
9           android:id="@+id/recyclerView"
10          android:layout_width="match_parent"
11          android:layout_height="match_parent"/>
12  </LinearLayout>
```

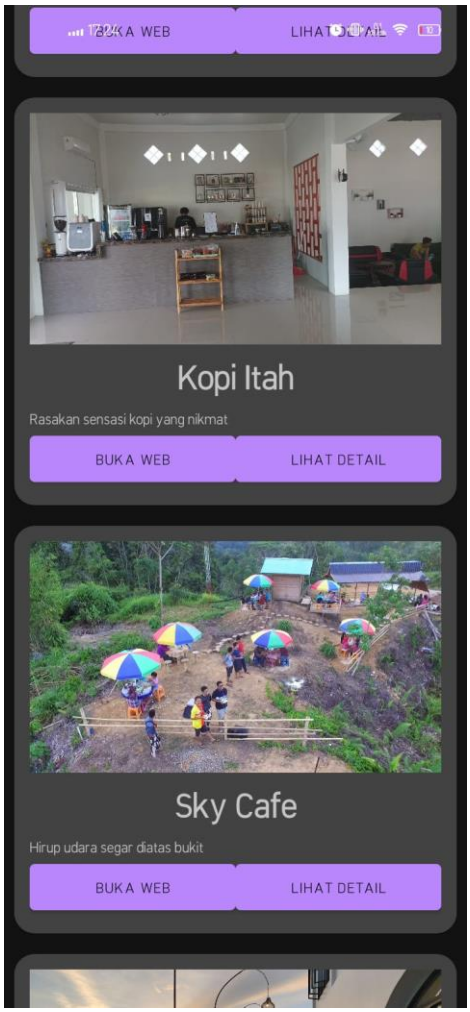
9. Fragment_detail.xml

Tabel 9. Source Code Jawaban Soal 1

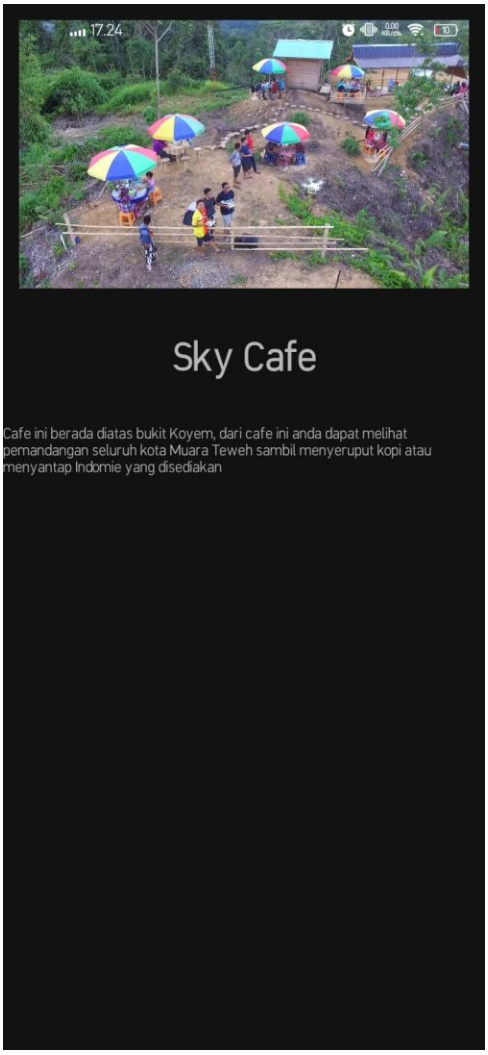
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       xmlns:app="http://schemas.android.com/apk/res-auto">
7       <androidx.constraintlayout.widget.ConstraintLayout
8           android:layout_width="match_parent"
9           android:layout_height="match_parent">
10      <ImageView
11          android:id="@+id/imageViewDetail"
12          android:layout_width="match_parent"
13          android:layout_height="200dp"
14          android:layout_margin="12dp"
15          android:scaleType="centerCrop"
16          android:layout_marginBottom="16dp"
17          app:layout_constraintTop_toTopOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintEnd_toEndOf="parent"/>
20      <TextView
21          android:id="@+id/titleViewDetail"
22          android:layout_width="match_parent"
23          android:layout_height="100dp"
24          android:gravity="center"
25          android:textStyle="bold"
26          android:text="detail"
27
28  app:layout_constraintTop_toBottomOf="@+id/imageViewDetail"
29          app:layout_constraintStart_toStartOf="parent"
30          app:layout_constraintEnd_toEndOf="parent"
31          android:textSize="36sp"/>
32      <TextView
33          android:id="@+id/textViewDetail"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:text="Detail"
37          android:layout_gravity="center"
```

```
38
39  app:layout_constraintTop_toBottomOf="@+id/titleViewDetail"
40          app:layout_constraintStart_toStartOf="parent"
41          app:layout_constraintEnd_toEndOf="parent"/>
42      </androidx.constraintlayout.widget.ConstraintLayout>
43  </FrameLayout>
```

B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1



Gambar 2. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt:

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace dari aplikasi. Selanjutnya, pada baris [2] hingga [4], berbagai library penting diimpor, termasuk komponen Android dan view binding yang diperlukan untuk mengelola tampilan. Pada baris [5], dideklarasikan kelas `MainActivity` yang merupakan turunan dari `AppCompatActivity`, yang berfungsi sebagai aktivitas utama aplikasi. Pada baris [6],

variabel `binding` dideklarasikan dengan tipe `ActivityMainBinding`, yang memungkinkan akses langsung ke elemen-elemen layout XML melalui view binding.

Pada baris [8-11], metode `onCreate()` diimplementasikan sebagai titik awal saat aktivitas dijalankan. Pada baris [9], `super.onCreate(savedInstanceState)` dipanggil untuk menjalankan inisialisasi dasar dari superclass. Di baris [10], `enableEdgeToEdge()` digunakan untuk mengaktifkan tampilan layar penuh (edge-to-edge display), memberikan pengalaman visual yang lebih modern. Kemudian, pada baris [11], layout di-inflate menggunakan `ActivityMainBinding.inflate(layoutInflater)` agar binding dapat digunakan. Akhirnya, pada baris [12], `setContentView(binding.root)` dipanggil untuk menampilkan root view dari layout sebagai tampilan utama aktivitas.

2. HomeFragment.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace dari fragment ini. Pada baris [2] hingga [8], sejumlah library penting diimpor untuk mendukung fungsionalitas fragment, seperti pengelolaan view, navigasi, dan RecyclerView. Pada baris [9], kelas `HomeFragment` dideklarasikan sebagai turunan dari `Fragment`, yang merepresentasikan salah satu tampilan (UI screen) dalam aplikasi.

Pada baris [10] dan [11], variabel `binding` dideklarasikan menggunakan pola nullable dengan properti `_binding`, serta getter non-null `binding` untuk akses yang aman. Pada baris [12], `viewModel` diinisialisasi menggunakan `by viewModels()` yang otomatis mengelola lifecycle sesuai fragment.

Pada baris [14–18], metode `onCreateView()` diimplementasikan untuk mengatur tampilan awal fragment. Di baris [16], layout di-inflate menggunakan `FragmentHomeBinding.inflate()`, dan hasil binding disimpan ke `_binding`. Kemudian, pada baris [17], root view dari binding dikembalikan agar fragment dapat menampilkan UI-nya.

Pada baris [20–23], metode `onViewCreated()` digunakan untuk inisialisasi lanjutan setelah view terbentuk. Pada baris [21], `LocationAdapter` diinisialisasi dengan data lokasi dari `viewModel` dan `NavController` untuk mendukung navigasi. Baris [22]

mengatur `LinearLayoutManager` pada `RecyclerView` agar menampilkan item secara vertikal, dan baris [23] menetapkan adapter ke `RecyclerView`.

Terakhir, pada baris [25–27], metode `onDestroyView()` bertugas membersihkan binding dengan menyetel `_binding` menjadi `null` agar tidak terjadi memory leak setelah view dihancurkan.

3. LocationViewModel.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai penanda namespace untuk kelas ini. Pada baris [2] dan [3], diimpor library yang dibutuhkan, termasuk `android.app.Application` dan `androidx.lifecycle.AndroidViewModel`, yang diperlukan untuk membuat ViewModel berbasis konteks aplikasi. Pada baris [4], kelas `LocationViewModel` dideklarasikan sebagai turunan dari `AndroidViewModel`, memungkinkan ViewModel mengakses `Application` context secara langsung.

Pada baris [5], konteks aplikasi diambil dari parameter konstruktor dan disimpan dalam variabel `context` untuk digunakan dalam pengambilan resource. Pada baris [7], fungsi `getLocations()` dideklarasikan, yang mengembalikan daftar `LocationModel`.

Pada baris [8–11], data berupa nama tempat, deskripsi, dan lokasi diambil dari resource `string-array` menggunakan `context.resources.getStringArray()`. Selanjutnya, pada baris [12–17], daftar gambar tempat disusun dalam bentuk list resource ID dari drawable.

Pada baris [19–25], dilakukan pemetaan data ke dalam objek `LocationModel` dengan menggunakan indeks array. Setiap elemen dalam list dibentuk dari gabungan elemen nama, deskripsi, lokasi, dan gambar berdasarkan posisi yang sama di masing-masing array, lalu dikembalikan sebagai list `LocationModel`.

4. LocationAdapter.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace utama untuk file ini. Pada baris [2–6], berbagai library penting diimpor, termasuk komponen RecyclerView, tampilan View, binding untuk layout, navigasi, dan Intent untuk membuka browser.

Pada baris [7], kelas `LocationAdapter` dideklarasikan sebagai turunan dari `RecyclerView.Adapter`, yang bertugas menampilkan daftar lokasi menggunakan ViewHolder.

Pada baris [9], inner class `LocationViewHolder` didefinisikan untuk memegang referensi view setiap item daftar, melalui view binding.

Pada baris [14–16], metode `onCreateViewHolder` diimplementasikan untuk meng-inflate layout XML dari item dan membungkusnya dalam `LocationViewHolder`.

Pada baris [18–35], metode `onBindViewHolder` digunakan untuk mengisi data dan menangani interaksi pengguna pada setiap item dalam daftar.

Pada baris [20–22], data dari objek `LocationModel` seperti nama, deskripsi, dan gambar dimasukkan ke dalam view-item masing-masing.

Pada baris [24–27], ketika tombol "browser" ditekan, akan dibuat `Intent` dengan `ACTION_VIEW` untuk membuka link lokasi di browser.

Pada baris [29–34], ketika tombol "detail" ditekan, `NavController` digunakan untuk navigasi ke fragment detail, membawa objek `LocationModel` sebagai argumen.

Pada baris [37], metode `getItemCount` mengembalikan jumlah total data dalam list, yang digunakan oleh RecyclerView untuk menentukan jumlah item yang perlu ditampilkan.

5. DetailFragment.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace untuk kelas ini. Pada baris [2–5], library-library yang diperlukan diimpor, termasuk komponen Fragment, View, dan binding.

Pada baris [6], kelas `DetailFragment` dideklarasikan sebagai turunan dari `Fragment`, yang berfungsi untuk menampilkan detail lokasi.

Pada baris [7–8], variabel binding didefinisikan untuk mengakses view dari layout dengan pola nullable dan non-null getter (`_binding` dan `binding`).

Pada baris [10–14], method `onCreateView` diimplementasikan untuk meng-inflate layout menggunakan `FragmentDetailBinding`, kemudian mengembalikan root view untuk ditampilkan.

Pada baris [16–23], method `onViewCreated` digunakan untuk mengambil objek `LocationModel` dari arguments (menggunakan `arguments?.getParcelable`), lalu data seperti nama, deskripsi, gambar, dan lokasi di-set ke komponen view yang sesuai.
Pada baris [25–27], method `onDestroyView` membersihkan referensi `_binding` untuk menghindari memory leak saat fragment dihancurkan.

6. LocationModel.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace untuk file ini.
Pada baris [3], `data class LocationModel` didefinisikan sebagai kelas data yang digunakan untuk merepresentasikan informasi lokasi nongki.
Pada baris [4–8], constructor utama dari `LocationModel` dideklarasikan dengan lima parameter:

- `nama` bertipe `String`, menyimpan nama tempat
- `fotoResId` bertipe `Int`, menyimpan ID resource gambar
- `deskripsi` bertipe `String`, menyimpan deskripsi tempat
- `rating` bertipe `String`, menyimpan nilai rating tempat
- `link` bertipe `String`, menyimpan URL lokasi atau informasi tambahan

7. Activity_main.xml

Pada baris [1], deklarasi versi XML dan encoding (`<?xml version="1.0" encoding="utf-8"?>`) ditulis untuk memastikan file diproses sebagai dokumen XML dengan encoding UTF-8.
Pada baris [2], elemen root berupa `androidx.fragment.app.FragmentContainerView` digunakan sebagai kontainer untuk menampilkan fragment secara dinamis.
Pada baris [3–5], namespace XML seperti `xmlns:android`, `xmlns:app`, dan `xmlns:tools` diimpor untuk mendukung atribut khusus Android dan AndroidX.
Pada baris [6], atribut `android:id` di-set ke `@+id/nav_host_fragment` untuk mengidentifikasi kontainer fragment ini dalam kode.
Pada baris [7], atribut `android:name` di-set ke

androidx.navigation.fragment.NavHostFragment yang berfungsi sebagai host untuk fragment navigasi.
Pada baris [8–9], android:layout_width dan android:layout_height diatur ke match_parent agar kontainer memenuhi seluruh layar.
Pada baris [10], tools:context menunjuk ke MainActivity, digunakan untuk pratinjau layout di Android Studio.
Pada baris [11], app:navGraph di-set ke @navigation/nav_graph, yaitu referensi ke file navigasi XML yang mendefinisikan alur navigasi aplikasi.
Pada baris [12], app:defaultNavHost di-set ke true agar NavHostFragment ini menjadi host utama untuk menangani aksi navigasi seperti tombol kembali.

8. Fragment_home.xml

Pada baris [1], terdapat deklarasi versi XML dan encoding (<?xml version="1.0" encoding="utf-8"?>) untuk memastikan file dibaca dengan benar sebagai dokumen XML berstandar UTF-8.
Pada baris [2], elemen root menggunakan LinearLayout yang berfungsi sebagai wadah vertikal atau horizontal bagi komponen UI di dalamnya.
Pada baris [3] dan [4], android:orientation di-set ke vertical untuk menata elemen secara vertikal, dan layout_width serta layout_height di-set ke match_parent agar memenuhi layar sepenuhnya.
Pada baris [6], elemen androidx.recyclerview.widget.RecyclerView digunakan untuk menampilkan daftar item secara efisien dalam jumlah besar.
Pada baris [7], android:id di-set ke @+id/recyclerView agar dapat diakses melalui kode Kotlin atau Java.
Pada baris [8] dan [9], layout_width dan layout_height untuk RecyclerView juga di-set ke match_parent, sehingga komponen ini mengisi seluruh ruang yang tersedia dalam LinearLayout.

9. Fragment_detail.xml

Pada baris [1], terdapat deklarasi XML <?xml version="1.0" encoding="utf-8"?> yang merupakan standar awal untuk mendefinisikan dokumen XML dengan encoding

UTF-8. Pada baris [2], elemen root dari layout ini adalah `FrameLayout`, sebuah container yang memungkinkan penumpukan elemen-elemen anak secara fleksibel. Pada baris [3] hingga [5], layout disiapkan dengan atribut `layout_width` dan `layout_height` yang di-set ke `match_parent`, serta definisi namespace Android (`xmlns:android`) dan `app` untuk mendukung ConstraintLayout.

Pada baris [7], `ConstraintLayout` digunakan sebagai child view utama dalam `FrameLayout`, karena layout ini mendukung penempatan elemen-elemen UI yang responsif berdasarkan constraint antar elemen. Layout ini digunakan untuk menyusun komponen UI seperti gambar, judul, dan deskripsi secara terstruktur.

Pada baris [9], terdapat `ImageView` yang digunakan untuk menampilkan gambar lokasi secara detail. Komponen ini diberi ID `@+id/imageViewDetail`, dengan tinggi layout `200dp`, margin `12dp` di sekelilingnya, dan properti `scaleType` diset ke `centerCrop` agar gambar dapat mengisi ruang yang tersedia tanpa distorsi. Posisi `ImageView` ini diletakkan di bagian atas layout dan diatur agar menempel ke sisi atas dan horizontal parent dengan menggunakan constraint.

Pada baris [18], `TextView` digunakan untuk menampilkan judul dari lokasi yang dipilih. Teks judul ditampilkan dengan gaya huruf tebal (`textStyle="bold"`), ukuran huruf besar (`textSize="36sp"`), dan rata tengah (`gravity="center"`) agar terlihat menonjol dan menarik perhatian pengguna. `TextView` ini diposisikan tepat di bawah `ImageView` menggunakan `layout_constraintTop_toBottomOf`.

Pada baris [27], terdapat `TextView` tambahan yang digunakan untuk menampilkan deskripsi dari lokasi secara lebih rinci. Komponen ini memiliki `layout_width wrap_content` agar lebarnya menyesuaikan dengan panjang teks, dan ditempatkan di bawah judul menggunakan constraint `layout_constraintTop_toBottomOf`. Dengan susunan ini, pengguna dapat melihat gambar, nama tempat, dan informasi deskriptif secara berurutan dan rapi di tampilan detail aplikasi.

SOAL 2

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

A. Pembahasan

RecyclerView memiliki performa yang tinggi, khususnya untuk menangani dataset yang besar. Meskipun bersifat boiler-plate yang panjang, pengembang memiliki kendali penuh atas UI. RecyclerView juga disupport dan digunakan secara luas oleh komunitas android.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/AllanoLintang/pemro-mobile