LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Firda Khoirunisa NIM. 2310817220025

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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Firda Khoirunisa
NIM : 2310817220025

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

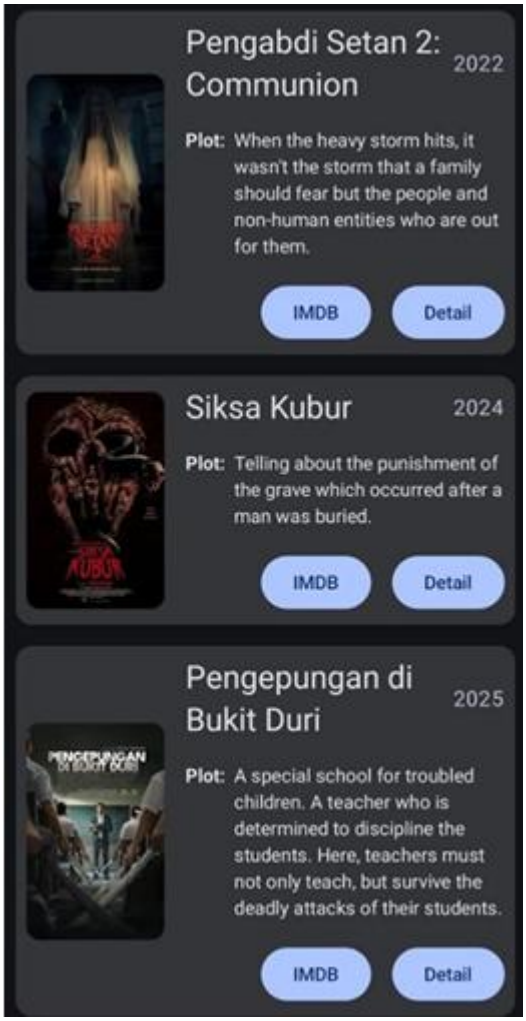
Soal Praktikum:

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

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

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

```
1  package com.example.modul3
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import com.example.modul3.databinding.ActivityMainBinding
6
7  class MainActivity : AppCompatActivity() {
8
9      private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         binding = ActivityMainBinding.inflate(layoutInflater)
14         setContentView(binding.root)
15
16         if (savedInstanceState == null) {
17             supportFragmentManager.beginTransaction()
18                 .replace(R.id.fragment_container,
19 HomeFragment())
20                 .commit()
21         }
22     }
23 }
```

Tabel 1. Source Code Jawaban Soal 1

2. Drama.kt

```
1  package com.example.modul3
2
3  import android.os.Parcelable
4  import kotlinx.parcelize.Parcelize
5
6  @Parcelize
7  data class Drama(
8      val title: String,
9      val year: String,
10     val genre: String,
11     val episodes: String,
12     val imageUrl: String,
13     val link: String
14 ) : Parcelable
```

Tabel 2. Source Code Jawaban Soal 1

3. ListDramaAdapter.kt

```
1  package com.example.modul3
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
```

```
import androidx.recyclerview.widget.RecyclerView
import com.bumptech.glide.Glide
import com.example.modul3.databinding.ItemDramaBinding

class ListDramaAdapter(
    private val listDrama: List<Drama>,
    private val onDetailClick: (Drama) -> Unit
) : RecyclerView.Adapter<ListDramaAdapter.MyViewHolder>() {

    inner class MyViewHolder(val binding: ItemDramaBinding) :
RecyclerView.ViewHolder(binding.root)

    override fun onCreateViewHolder(parent: ViewGroup, viewType:
Int): MyViewHolder {
        val binding =
ItemDramaBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return MyViewHolder(binding)
    }

    override fun getItemCount(): Int = listDrama.size

    override fun onBindViewHolder(holder: MyViewHolder,
position: Int) {
        val drama = listDrama[position]
        holder.binding.drama = drama

        Glide.with(holder.binding.root.context)
            .load(drama.imageUrl)
            .into(holder.binding.imageView)

        holder.binding.buttonDetail.setOnClickListener {
            onDetailClick(drama)
        }
    }
}
```

Tabel 3. Source Code Jawaban Soal 1

4. HomeFragment.kt

```
package com.example.modul3

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.modul3.databinding.FragmentHomeBinding

class HomeFragment : Fragment(R.layout.fragment_home) {
    private lateinit var binding: FragmentHomeBinding
```

```
    private lateinit var dramaAdapter: ListDramaAdapter

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentHomeBinding.inflate(inflater,
container, false)

        val dramas = listOf(
            Drama(
                "Mysterious Lotus Casebook",
                "2023",
                "Mystery, Wuxia",
                "40 Episodes",
                "https://i.mydramalist.com/kAozjj_4c.jpg?v=1",
                "https://www.iq.com/album/mysterious-lotus-
casebook-2023-hg4cefqzed?lang=en_us"
            ),
            Drama(
                "Love Game In Eastern Fantasy",
                "2024",
                "Romance, Wuxia, Fantasy",
                "32 Episodes",
                "https://i.mydramalist.com/VXOLzy_4c.jpg?v=1",

"https://wetv.vip/en/play/227hqhmxvabwggz/d4100tnphtz"
            ),
            Drama(
                "Melody of Golden Age",
                "2024",
                "Historical, Mystery, Romance, Drama",
                "40 Episodes",
                "https://i.mydramalist.com/d0Q62d_4c.jpg?v=1",
                "https://www.iq.com/album/melody-of-golden-age-
2025-vobwm47vvd?lang=en_us"
            ),
            Drama(
                "One And Only",
                "2021",
                "Military, Historical, Romance, Political",
                "24 Episodes",
                "https://i.mydramalist.com/BLrkR_4c.jpg?v=1",
                "https://www.iq.com/album/one-and-only-2021-
24mppdujjp9?lang=en_us"
            ),
            Drama(
                "Go Ahead",
                "2020",
                "Comedy, Romance, Drama, Melodrama",
                "46 Episodes",
                "https://i.mydramalist.com/exXvQ_4c.jpg?v=1",
```

```
                "https://www.iq.com/album/go-ahead-2020-
1bm24k36pk9?lang=en_us"
            )
        )

        dramaAdapter = ListDramaAdapter(dramas) { selectedDrama
->
            val fragment = DetailFragment()
            val bundle = Bundle()
            bundle.putParcelable("drama", selectedDrama)
            fragment.arguments = bundle

            parentFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, fragment)
                .addToBackStack(null)
                .commit()
        }

        binding.recyclerView.apply {
            layoutManager = LinearLayoutManager(context)
            adapter = dramaAdapter
        }
        return binding.root
    }
}
```

Tabel 4. Source Code Jawaban Soal 1

5. DetailFragment.kt

```
package com.example.modul3

import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import com.example.modul3.databinding.FragmentDetailBinding

class DetailFragment : Fragment() {
    private lateinit var binding: FragmentDetailBinding

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        binding = FragmentDetailBinding.inflate(inflater,
container, false)

        val drama = arguments?.getParcelable<Drama>("drama")
        drama?.let { selectedDrama ->
```

```
24              binding.drama            =            selectedDrama
25
26              binding.openLinkButton.setOnClickListener         {
27                  val    intent    =    Intent(Intent.ACTION_VIEW,
28  Uri.parse(selectedDrama.link))
29                  startActivity(intent)
30              }
31
32              binding.backButton.setOnClickListener             {
33
34  requireActivity().supportFragmentManager.popBackStack()
35              }
36          }
37
38          return                                          binding.root
39      }
40  }
```

Tabel 5. Source Code Jawaban Soal 1

6. activity_main.xml

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

Tabel 6. Source Code Jawaban Soal 1

7. item_drama.xml

```
1   <?xml              version="1.0"              encoding="utf-8"?>
2   <layout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto">
5
6       <data>
7           <variable
8               name="drama"
9               type="com.example.modul3.Drama"                   />
10      </data>
11
12      <androidx.cardview.widget.CardView
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
```

```
15          android:layout_margin="8dp"
16          android:padding="8dp"
17          android:elevation="4dp"
18          android:clickable="true"
19          android:focusable="true">
20
21          <androidx.constraintlayout.widget.ConstraintLayout
22              android:layout_width="match_parent"
23              android:layout_height="wrap_content">
24
25              <ImageView
26                  android:id="@+id/imageView"
27                  android:layout_width="0dp"
28                  android:layout_height="0dp"
29                  android:layout_marginEnd="8dp"
30
31  android:contentDescription="@string/detail_image"
32                  android:scaleType="centerCrop"
33                  app:layout_constraintEnd_toEndOf="parent"
34                  app:layout_constraintHeight_default="percent"
35                  app:layout_constraintHeight_percent="0.5"
36                  app:layout_constraintHorizontal_bias="0.0"
37                  app:layout_constraintStart_toStartOf="parent"
38                  app:layout_constraintTop_toTopOf="parent"    />
39
40              <TextView
41                  android:id="@+id/titleText"
42                  android:layout_width="wrap_content"
43                  android:layout_height="wrap_content"
44                  android:text="@{drama.title}"
45                  android:textSize="16sp"
46                  app:layout_constraintEnd_toEndOf="parent"
47                  app:layout_constraintStart_toStartOf="parent"
48
49  app:layout_constraintTop_toBottomOf="@id/imageView"          />
50
51              <TextView
52                  android:id="@+id/yearText"
53                  android:layout_width="wrap_content"
54                  android:layout_height="wrap_content"
55                  android:text="@{drama.year}"
56                  android:textSize="14sp"
57                  app:layout_constraintEnd_toEndOf="parent"
58                  app:layout_constraintStart_toStartOf="parent"
59
60  app:layout_constraintTop_toBottomOf="@id/titleText"          />
61
62              <Button
63                  android:id="@+id/buttonDetail"
64                  android:layout_width="wrap_content"
65                  android:layout_height="wrap_content"
66                  android:text="Detail"
```

```
                android:layout_marginTop="8dp"

app:layout_constraintTop_toBottomOf="@id/yearText"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"/>

        </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
</layout>
```

Tabel 7. Source Code Jawaban Soal 1

8. fragment_home.xml

```
<?xml              version="1.0"              encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"            />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 8. Source Code Jawaban Soal 1

9. fragment_detail.xml

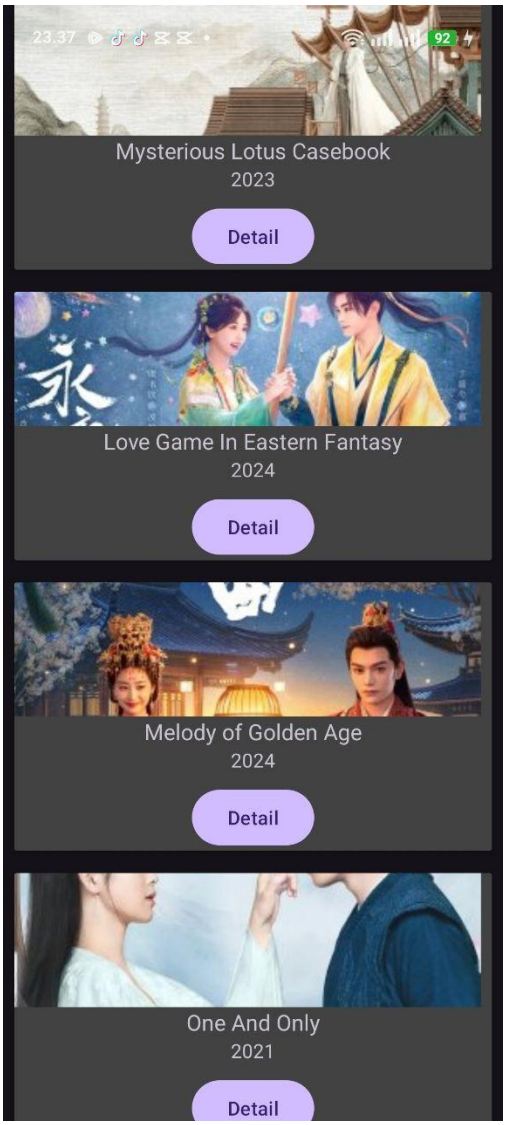
```
<?xml              version="1.0"              encoding="utf-8"?>
<layout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <data>
        <variable
            name="drama"
            type="com.example.modul3.Drama"                  />
    </data>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".DetailFragment">

        <LinearLayout
            android:orientation="vertical"
```

```
            android:padding="16dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal">

            <ImageView
                android:id="@+id/detail_image"
                android:layout_width="match_parent"
                android:layout_height="500dp"
                android:scaleType="centerCrop"

android:contentDescription="@string/detail_image"
                app:imageUrl="@{drama.imageUrl}"             />

            <TextView
                android:id="@+id/title_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@{drama.title}"
                android:textSize="20sp"
                android:textStyle="bold"
                android:paddingTop="8dp"
                android:layout_gravity="center_horizontal"/>

            <TextView
                android:id="@+id/year_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@{drama.year}"
                android:textSize="16sp"
                android:paddingTop="4dp"
                android:layout_gravity="center_horizontal"/>

            <TextView
                android:id="@+id/genre_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@{drama.genre}"
                android:textSize="16sp"
                android:paddingTop="4dp"
                android:layout_gravity="center_horizontal"/>

            <TextView
                android:id="@+id/episodes_text"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@{drama.episodes}"
                android:textSize="16sp"
                android:paddingTop="4dp"
                android:layout_gravity="center_horizontal"/>

            <Button
```

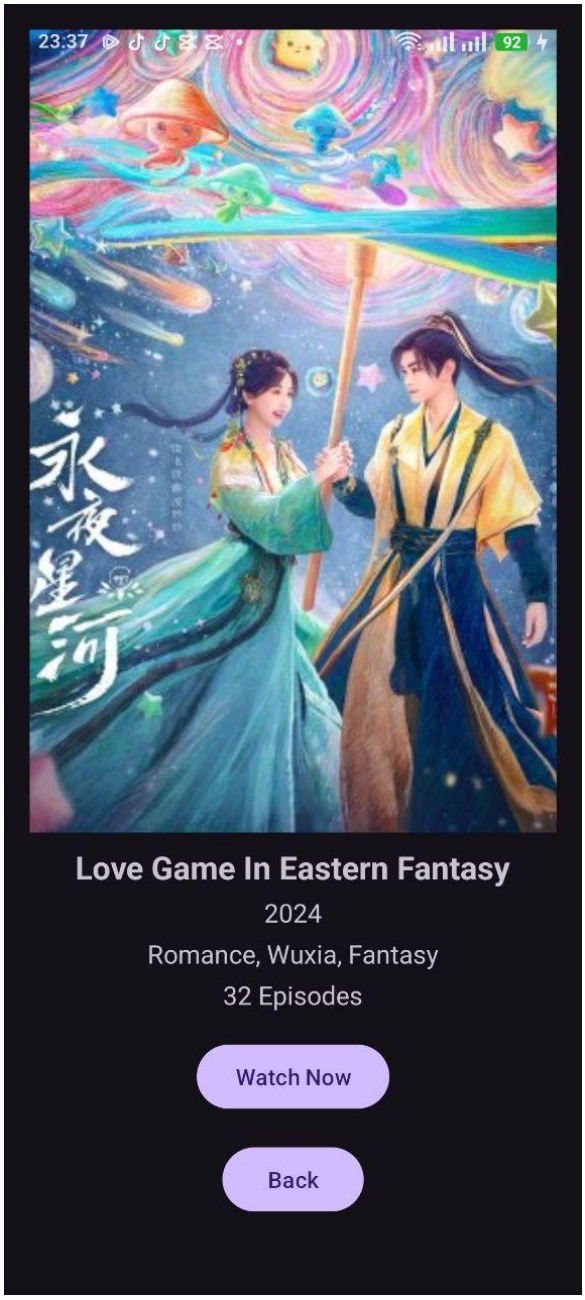
```
72                  android:id="@+id/openLinkButton"
73                  android:layout_width="wrap_content"
74                  android:layout_height="wrap_content"
75                  android:text="Watch                          Now"
76                  android:layout_marginTop="16dp"
77                  android:layout_gravity="center_horizontal"   />
78
79              <Button
80                  android:id="@+id/backButton"
81                  android:layout_width="wrap_content"
82                  android:layout_height="wrap_content"
83                  android:text="Back"
84                  android:layout_marginTop="16dp"
85                  android:layout_gravity="center_horizontal"/>
86          </LinearLayout>
87      </ScrollView>
88  </layout>
```

Tabel 9. Source Code Jawaban Soal 1

B. Output Program



Gambar 3. Screenshot Hasil Jawaban Soal 1



Gambar 4. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt:

File ini merupakan titik awal aplikasi Android, tempat di mana layout utama ditentukan dan fragment pertama dimuat. Di dalam metode onCreate(), fungsi setContentView() digunakan untuk menetapkan layout XML utama (activity_main.xml) yang berisi FrameLayout sebagai wadah fragment. Selanjutnya, ada pemeriksaan terhadap savedInstanceState, yang digunakan untuk memastikan bahwa fragment HomeFragment hanya ditambahkan ketika aplikasi pertama kali dijalankan, bukan saat dipulihkan dari keadaan sebelumnya. Jika kondisinya terpenuhi (yakni savedInstanceState == null), maka fragment HomeFragment dimuat ke dalam fragment_container menggunakan supportFragmentManager. Hal ini memastikan pengguna langsung melihat daftar drama saat aplikasi dibuka, dan fragment tidak dimuat ulang secara tidak perlu saat orientasi layar berubah atau aplikasi dikembalikan dari latar belakang.

2. Drama.kt

File ini berisi sebuah data class bernama Drama, yang digunakan sebagai model data dalam aplikasi. Data class ini menyimpan lima properti utama: title (judul drama), yearGenre (kombinasi tahun rilis dan genre), episodes (jumlah episode), imageUrl (alamat URL gambar poster drama), dan link (URL video atau halaman nonton). Dengan adanya data class ini, setiap objek drama dapat dengan mudah dibuat dan dikelola di dalam daftar. Karena Drama mengimplementasikan Parcelable, objek-objek ini bisa dengan mudah dikirimkan antar fragment melalui Bundle, yang sangat penting dalam navigasi aplikasi.

3. ListDramaAdapter.kt

File ini berperan sebagai jembatan antara data dan tampilan pada RecyclerView. Kelas DramaAdapter mengambil daftar objek Drama serta sebuah fungsi callback bernama onDetailClick yang akan dipanggil saat pengguna menekan tombol "Detail". Di dalamnya terdapat kelas DramaViewHolder, yang mengikat data ke layout item_drama.xml menggunakan data binding. Pada metode onBindViewHolder, adapter akan menetapkan data Drama ke properti binding.drama dan juga menetapkan listener tombol "Detail". Saat tombol

ditekan, fungsi onDetailClick akan dijalankan dengan parameter objek Drama terkait, sehingga aktivitas atau fragment dapat merespons aksi pengguna untuk menampilkan detail. Struktur ini memisahkan logika tampilan dan logika interaksi, membuat kode lebih modular dan mudah dirawat.

4. HomeFragment.kt

Fragment ini bertugas menampilkan daftar drama yang tersedia kepada pengguna. Pada saat onCreateView(), fragment ini menggunakan FragmentHomeBinding untuk menghubungkan layout fragment_home.xml dengan kode Kotlin. RecyclerView yang ada di layout kemudian dihubungkan dengan DramaAdapter, dan daftar drama diisi menggunakan fungsi getListDrama() yang berisi data dummy. Fungsi ini menciptakan beberapa objek Drama secara manual dan menambahkannya ke dalam list. Salah satu bagian penting di sini adalah penanganan tombol "Detail" pada setiap item. Saat tombol tersebut ditekan, aplikasi akan membuat instance DetailFragment, mengirim data drama melalui Bundle, dan mengganti tampilan fragment saat ini dengan DetailFragment. Proses ini dilakukan menggunakan parentFragmentManager dan transaksi fragment, memungkinkan navigasi antar halaman tanpa perlu memulai aktivitas baru.

5. DetailFragment.kt

Fragment ini digunakan untuk menampilkan informasi lengkap dari drama yang dipilih oleh pengguna. Di dalam metode onCreateView(), data Drama diterima dari arguments menggunakan metode getParcelable(), yang sebelumnya dikirim dari HomeFragment. Data ini kemudian diikat ke layout menggunakan FragmentDetailBinding, sehingga semua elemen UI seperti judul, genre, jumlah episode, dan gambar secara otomatis terisi dengan data yang sesuai. Fragment ini juga menyediakan dua aksi utama bagi pengguna. Pertama, tombol "Watch Now" yang menggunakan Intent.ACTION_VIEW untuk membuka link video di browser atau aplikasi terkait. Kedua, tombol "Back" yang memungkinkan pengguna kembali ke daftar drama dengan memanggil requireActivity().onBackPressedDispatcher.onBackPressed(), menggantikan fungsi popBackStack() agar tetap kompatibel dengan berbagai versi Android.

6. activity_main.xml

File ini adalah layout utama untuk aplikasi, yang menggunakan ConstraintLayout sebagai root view. Layout ini hanya memiliki satu elemen, yaitu FrameLayout dengan ID fragment_container, yang akan menjadi wadah untuk menampilkan fragment lainnya. FrameLayout ini mengisi seluruh layar dan akan digunakan untuk menampung fragment yang akan di-embed dalam activity ini. Jadi, layout utama aplikasi hanya berfokus pada tempat untuk memuat fragment, tanpa ada elemen tampilan lainnya.

7. item_drama.xml

File ini mendesain tampilan item dalam daftar drama yang akan ditampilkan di dalam RecyclerView. Pada bagian awal, dideklarasikan sebuah variabel drama yang bertipe com.example.modul3.Drama menggunakan data binding. Di dalam layout, terdapat sebuah CardView dengan margin, padding, dan elevasi untuk memberikan efek bayangan. Di dalam CardView, terdapat beberapa elemen tampilan: sebuah ImageView untuk menampilkan gambar drama dengan proporsi yang diatur agar terlihat pas, dan dua TextView untuk menampilkan judul dan tahun dari drama. Selain itu, ada sebuah tombol "Detail" yang digunakan untuk menunjukkan detail dari drama ketika diklik. Semua elemen ini diletakkan dengan bantuan ConstraintLayout, yang memastikan elemen-elemen tersebut terorganisir dengan baik di dalam layout.

8. fragment_home.xml

File ini adalah layout untuk fragment yang menampilkan daftar drama dalam bentuk RecyclerView. RecyclerView diatur untuk mengisi seluruh ukuran fragment, dari atas ke bawah dan kiri ke kanan, berkat penggunaan ConstraintLayout dan pengaturan constraint yang tepat. Fragment ini bertugas untuk menampilkan daftar item dalam aplikasi, di mana setiap item tersebut akan menggunakan layout item_drama.xml yang telah didefinisikan sebelumnya. Dengan demikian, RecyclerView akan menjadi tempat di mana pengguna dapat menggulir dan melihat berbagai drama yang tersedia di aplikasi.

9. fragment_detail.xml

File ini digunakan untuk menampilkan detail dari sebuah drama yang dipilih. Layout menggunakan ScrollView untuk memastikan bahwa konten dapat digulir jika ukurannya melebihi layar. Di dalam ScrollView, terdapat LinearLayout dengan orientasi vertikal untuk mengatur elemen-elemen yang ada. Elemen pertama adalah ImageView yang menampilkan gambar drama, yang dihubungkan dengan URL gambar melalui data binding. Berikutnya ada beberapa TextView yang menampilkan informasi tentang drama, seperti judul, tahun, genre, dan jumlah episode, semuanya diatur agar terpusat secara horizontal dan diberi padding agar lebih rapi. Di bagian bawah, terdapat dua tombol: satu untuk membuka tautan ke video drama (dengan teks "Watch Now"), dan satu lagi untuk kembali ke halaman sebelumnya (dengan teks "Back"). Semua elemen ini diatur dengan cara yang membuat tampilan detail drama terlihat terstruktur dan mudah dinavigasi.

Jawaban Soal 2

RecyclerView masih banyak digunakan karena memberikan kontrol dan fleksibilitas tinggi untuk menampilkan data kompleks. Komponen ini mendukung berbagai fitur seperti animasi, penggunaan LayoutManager yang dapat disesuaikan, serta efisiensi memori melalui penggunaan ViewHolder. Meskipun lebih rumit dan memerlukan kode tambahan seperti adapter dan view holder, RecyclerView cocok untuk proyek yang berbasis View XML dan memerlukan kustomisasi tingkat lanjut. Sementara itu, LazyColumn dari Jetpack Compose menawarkan cara yang lebih sederhana dan deklaratif untuk menampilkan daftar. Penulisan kode lebih ringkas tanpa perlu membuat adapter, sehingga cocok untuk pengembangan modern berbasis Compose. Namun, LazyColumn memiliki keterbatasan dalam hal fleksibilitas dan kustomisasi jika dibandingkan dengan RecyclerView. Pilihan antara keduanya bergantung pada pendekatan UI yang digunakan dalam proyek.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

Pemrograman-Mobile/MODUL3 at main · firdakhrns/Pemrograman-Mobile