LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

OLEH:

ZAHRA NABILA **NIM 2310817320007**

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MEI 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 : Zahra Nabila
NIM : 2310817320007

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 19881027 201903 20 13

DAFTAR ISI

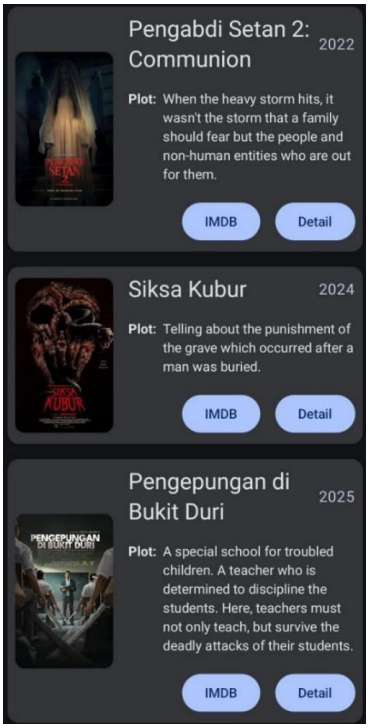
DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

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

File MainActivity.kt

```
1  package com.example.recycler_view
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5
6  class MainActivity : AppCompatActivity() {
7      override fun onCreate(savedInstanceState: Bundle?) {
8          super.onCreate(savedInstanceState)
9          setContentView(R.layout.activity_main)
10
11         val fragmentManager = supportFragmentManager
12         val homeFragment = HomeFragment()
13         val fragment =
   fragmentManager.findFragmentByTag(HomeFragment::class.java
   .simpleName)
14         if (fragment !is HomeFragment) {
15             fragmentManager
16                 .beginTransaction()
17                 .add(R.id.frame_container, homeFragment,
18 HomeFragment::class.java.simpleName)
                   .commit()
19         }
20     }
21 }
```

Table 1. Source Code Jawaban Soal 1

File DetailFragment.kt

```
1  package com.example.recycler_view
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.recycler_view.databinding.FragmentDetailBinding
9
10 class DetailFragment : Fragment() {
11
12     private var _binding: FragmentDetailBinding? = null
13     private val binding get() = _binding!!
14
15     override fun onCreateView(
16         inflater: LayoutInflater, container: ViewGroup?,
17         savedInstanceState: Bundle?
18  ): View? {
19         _binding = FragmentDetailBinding.inflate(inflater,
   container, false)
20
21         val title = arguments?.getString("EXTRA_TITLE")
22         val photo = arguments?.getInt("EXTRA_PHOTO")
23         val plot = arguments?.getString("EXTRA_PLOT")
24         val year = arguments?.getString("EXTRA_YEAR")
25         val cast = arguments?.getString("EXTRA_CAST")
26
27         binding.tvName.text = title
28         binding.tvPlot.text = plot
29         binding.tvYear.text = "$year"
30         binding.tvCast.text = "$cast"
31         photo?.let {
```

```
    binding.imgItemPhoto.setImageResource(it) }
32
33          return binding.root
34      }
35
36      override fun onDestroyView() {
37          super.onDestroyView()
38          _binding = null
39      }
40  }
```

Table 2. Source Code Jawaban Soal 1

File HomeFragment.kt

```
1   package com.example.recycler_view
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import androidx.recyclerview.widget.LinearLayoutManager
11  import
    com.example.recycler_view.databinding.FragmentHomeBinding
12
13  class HomeFragment : Fragment() {
14
15      private var _binding: FragmentHomeBinding? = null
16      private val binding get() = _binding!!
17
```

```
18      override fun onCreateView(
19          inflater: LayoutInflater, container: ViewGroup?,
20          savedInstanceState: Bundle?
21      ): View? {
22          _binding = FragmentHomeBinding.inflate(inflater,
    container, false)
23          return binding.root
24      }
25
26      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
27          super.onViewCreated(view, savedInstanceState)
28          binding.rvDrama.layoutManager =
    LinearLayoutManager(requireContext())
29          binding.rvDrama.setHasFixedSize(true)
30
31          val dramaAdapter = DramaAdapter(getDramaList(),
32              onWikiClick = { link ->
33                  val uri = Uri.parse(link)
34                  startActivity(Intent(Intent.ACTION_VIEW,
    uri))
35              },
36              onDetailClick = { title, photo, plot, year,
    cast ->
37                  val detailFragment =
    DetailFragment().apply {
38                      arguments = Bundle().apply {
39                          putString("EXTRA_TITLE", title)
40                          putInt("EXTRA_PHOTO", photo)
41                          putString("EXTRA_PLOT", plot)
42                          putString("EXTRA_YEAR", year)
```

```
43                          putString("EXTRA_CAST", cast)
44                      }
45                  }
46                  parentFragmentManager.beginTransaction()
47                      .replace(R.id.frame_container,
    detailFragment)
48                      .addToBackStack(null)
49                      .commit()
50              })
51
52          binding.rvDrama.adapter = dramaAdapter
53      }
54
55      private fun getDramaList(): ArrayList<Series> {
56          val                  dataTitle                  =
    resources.getStringArray(R.array.data_name)
57          val                  dataLink                   =
    resources.getStringArray(R.array.data_link)
58          val                  dataPhoto                  =
    resources.obtainTypedArray(R.array.data_photo)
59          val                  dataPlot                   =
    resources.getStringArray(R.array.data_plot)
60          val                  dataYear                   =
    resources.getStringArray(R.array.data_year)
61          val                  dataCast                   =
    resources.getStringArray(R.array.data_cast)
62          val listDrama = ArrayList<Series>()
63          for (i in dataTitle.indices) {
64              val drama = Series(
65                  dataTitle[i],
66                  dataLink[i],
```

```
                dataPhoto.getResourceId(i, -1),
                dataPlot[i],
                dataYear[i],
                dataCast[i]
            )
            listDrama.add(drama)
        }
        dataPhoto.recycle()
        return listDrama
    }

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

Table 3. Source Code Jawaban Soal 1

File ListSeriesAdapter.kt

```
package com.example.recycler_view

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class DramaAdapter(
    private val listDrama: ArrayList<Series>,
```

```
13      private val onWikiClick: (String) -> Unit,
14      private val onDetailClick: (String, Int, String,
    String, String) -> Unit
15  ) : RecyclerView.Adapter<DramaAdapter.ViewHolder>() {
16
17      class ViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
18          val imgPhoto: ImageView =
    itemView.findViewById(R.id.img_item_photo)
19          val tvTitle: TextView =
    itemView.findViewById(R.id.tv_item_name)
20          val tvPlot: TextView =
    itemView.findViewById(R.id.tv_item_plot)
21          val btnWiki: Button =
    itemView.findViewById(R.id.btn_wiki)
22          val btnDetail: Button =
    itemView.findViewById(R.id.button_detail)
23      }
24
25      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ViewHolder {
26          val view: View =
    LayoutInflater.from(parent.context).inflate(R.layout.item
    _list, parent, false)
27          return ViewHolder(view)
28      }
29
30      override fun getItemCount(): Int = listDrama.size
31
32      override fun onBindViewHolder(holder: ViewHolder,
    position: Int) {
```

```
33          val (title, link, photo, plot, year, cast) =
    listDrama[position]
34          holder.tvTitle.text = title
35          holder.imgPhoto.setImageResource(photo)
36          holder.tvPlot.text = "$cast"
37          holder.btnWiki.setOnClickListener {
    onWikiClick(link) }
38          holder.btnDetail.setOnClickListener {
    onDetailClick(title, photo, plot, year, cast) }
39      }
40  }
```

Table 4. Source Code Jawaban Soal 1

File Series.kt

```
1   package com.example.recycler_view
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5
6   @Parcelize
7   data class Series(
8       val title: String,
9       val link: String,
10      val photo: Int,
11      val plot: String,
12      val year: String,
        val cast: String
13  ) : Parcelable
14
```

Table 5. Source Code Jawaban Soal 1

File activity_main.xml

```
1   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
2       xmlns:app="http://schemas.android.com/apk/res-auto"
3       xmlns:tools="http://schemas.android.com/tools"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       tools:context=".MainActivity">
7
8       <FrameLayout
9           android:id="@+id/frame_container"
10          android:layout_width="match_parent"
11          android:layout_height="match_parent" />
12  </androidx.constraintlayout.widget.ConstraintLayout>
```

Table 6. Source Code Jawaban Soal 1

File fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android
    "
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:fillViewport="true"
8       android:background="@drawable/bg"
9       tools:context=".DetailFragment">
10
11      <androidx.constraintlayout.widget.ConstraintLayout
```

```
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:paddingBottom="32dp">
15
16          <ImageView
17              android:id="@+id/img_item_photo"
18              android:layout_width="341dp"
19              android:layout_height="458dp"
20              android:layout_marginTop="32dp"
21              android:scaleType="centerCrop"
22              app:layout_constraintEnd_toEndOf="parent"
23              app:layout_constraintStart_toStartOf="parent"
24              app:layout_constraintTop_toTopOf="parent"
25              tools:src="@drawable/antares" />
26
27          <TextView
28              android:id="@+id/tv_name"
29              android:layout_width="wrap_content"
30              android:layout_height="wrap_content"
31              android:layout_marginTop="16dp"
                android:fontFamily="serif"
32              android:textSize="25sp"
33              android:textStyle="bold"
34              app:layout_constraintEnd_toEndOf="parent"
35              app:layout_constraintStart_toStartOf="parent"
36               app:layout_constraintTop_toBottomOf="@id/img_
    item_photo"
37              tools:text="Judul Series" />
38
39          <TextView
40              android:id="@+id/tv_year"
```

```
41              android:layout_width="wrap_content"
42              android:layout_height="wrap_content"
43              android:text="Tahun: "
44              android:fontFamily="serif"
45              android:textSize="18sp"
46              android:layout_marginTop="8dp"
47               app:layout_constraintTop_toBottomOf="@id/tv_n
    ame"
48              app:layout_constraintStart_toStartOf="parent"
49              app:layout_constraintEnd_toEndOf="parent" />
50
51          <TextView
52              android:id="@+id/tv_cast"
53              android:layout_width="0dp"
54              android:layout_height="wrap_content"
55              android:text="Pemeran: "
56              android:fontFamily="serif"
57              android:textSize="18sp"
58              android:layout_marginTop="8dp"
59              android:layout_marginHorizontal="16dp"
60              android:textAlignment="center"
61               app:layout_constraintTop_toBottomOf="@id/tv_y
    ear"
62              app:layout_constraintStart_toStartOf="parent"
63              app:layout_constraintEnd_toEndOf="parent" />
64
65          <TextView
66              android:id="@+id/tv_plot"
67              android:layout_width="0dp"
68              android:layout_height="wrap_content"
69              android:layout_margin="16dp"
```

```
70              android:fontFamily="serif"
71              android:text="Sinopsis: "
72              android:textAlignment="center"
73              android:textSize="18sp"
74               app:layout_constraintBottom_toBottomOf="paren
    t"
75              app:layout_constraintEnd_toEndOf="parent"
76              app:layout_constraintStart_toStartOf="parent"
77
78  app:layout_constraintTop_toBottomOf="@+id/tv_cast" />
79
80      </androidx.constraintlayout.widget.ConstraintLayout>
81  </ScrollView>
```

Table 7. Source Code Jawaban Soal 1

File fragment_home.xml

```
1   <?xml           version="1.0"           encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@drawable/bg2">
8
9       <androidx.recyclerview.widget.RecyclerView
10          android:id="@+id/rvDrama"
11          android:layout_width="0dp"
12          android:layout_height="0dp"
13          app:layout_constraintTop_toTopOf="parent"
14          app:layout_constraintBottom_toBottomOf="parent"
15          app:layout_constraintStart_toStartOf="parent"
```

```
16          app:layout_constraintEnd_toEndOf="parent"       />
17  </androidx.constraintlayout.widget.ConstraintLayout>
```

Table 8. Source Code Jawaban Soal 1

File item_list.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android
    "
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="match_parent"
5       android:layout_height="wrap_content"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_margin="8dp"
8       app:cardCornerRadius="8dp"
9       app:cardElevation="4dp">
10
11      <androidx.constraintlayout.widget.ConstraintLayout
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:padding="8dp"
15          tools:src="@drawable/bg2">
16
17          <ImageView
18              android:id="@+id/img_item_photo"
19              android:layout_width="70dp"
20              android:layout_height="100dp"
21              android:scaleType="centerCrop"
22              app:layout_constraintStart_toStartOf="parent"
23              app:layout_constraintTop_toTopOf="parent"
```

```
24              app:layout_constraintBottom_toBottomOf="parent
"
25              tools:src="@drawable/antares" />
26
27          <TextView
28              android:id="@+id/tv_item_name"
29              android:layout_width="0dp"
30              android:layout_height="wrap_content"
31              android:layout_marginStart="12dp"
32              android:textSize="18sp"
33              android:textStyle="bold"
34              android:fontFamily="serif"
35              android:textColor="#000000"
36             app:layout_constraintStart_toEndOf="@id/img_it
em_photo"
37              app:layout_constraintTop_toTopOf="parent"
38              app:layout_constraintEnd_toEndOf="parent"
39              tools:text="Antares" />
40
41          <TextView
42              android:id="@+id/tv_item_plot"
43              android:layout_width="0dp"
44              android:layout_height="wrap_content"
45              android:layout_marginTop="4dp"
46              android:textSize="14sp"
47              android:fontFamily="serif"
48             app:layout_constraintTop_toBottomOf="@id/tv_it
em_name"
49              app:layout_constraintStart_toStartOf="@id/tv_i
tem_name"
50              app:layout_constraintEnd_toEndOf="parent"
```

```
51              tools:text="Cast: " />
52
53          <Button
54              android:id="@+id/btn_wiki"
55              android:layout_width="wrap_content"
56              android:layout_height="wrap_content"
57              android:layout_marginStart="60dp"
58              android:layout_marginTop="12dp"
59              android:backgroundTint="#9bc6ec"
60              android:fontFamily="serif"
61               android:textStyle="bold"
62              android:text="WeTV"
63             app:layout_constraintStart_toStartOf="@id/tv_i
    tem_plot"
64             app:layout_constraintTop_toBottomOf="@id/tv_it
    em_plot" />
65
66          <Button
67              android:id="@+id/button_detail"
68              android:layout_width="wrap_content"
69              android:layout_height="wrap_content"
70              android:layout_marginStart="40dp"
71              android:backgroundTint="#9bc6ec"
72              android:fontFamily="serif"
73              android:textStyle="bold"
74              android:text="Detail"
75             app:layout_constraintStart_toEndOf="@id/btn_wi
    ki"
76             app:layout_constraintTop_toTopOf="@id/btn_wiki
    " />
77
```

```
78      </androidx.constraintlayout.widget.ConstraintLayout>
79  </androidx.cardview.widget.CardView>
```

Table 9. Source Code Jawaban Soal 1

File colors.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3       <color name="black">#FF000000</color>
4       <color name="white">#FFFFFFFF</color>
5       <color name="blue">#9bc6ec</color>
6   </resources>
```

Table 10. Source Code Jawaban Soal 1

File strings.xml

```
1   <resources>
2       <string name="app_name">Rycycler View</string>
3       <string name="btn_detail">Detail</string>
4       <string name="btn_wiki">Wiki</string>
5
6       <string-array name="data_name">
7           <item>Antares</item>
8           <item>Harus Kawin</item>
9           <item>Kaget Nikah</item>
10          <item>Jodoh atau Bukan</item>
11          <item>Kisah untuk Geri</item>
12          <item>Mozachiko</item>
13          <item>5 Detik dan Rasa Rindu</item>
14          <item>17 Selamanya</item>
15          <item>Teluk Alaska</item>
16          <item>2 Wajah Arjuna</item>
17      </string-array>
```

```
18
19      <string-array name="data_photo">
20          <item>@drawable/antares</item>
21          <item>@drawable/harus_kawin</item>
22          <item>@drawable/kaget_nikah</item>
23          <item>@drawable/jodoh_bukan</item>
24          <item>@drawable/kisah_geri</item>
25          <item>@drawable/mozachiko</item>
26          <item>@drawable/rasa_rindu</item>
27          <item>@drawable/selamanya</item>
28          <item>@drawable/teluk_alaska</item>
29          <item>@drawable/wajah_arjuna</item>
30      </string-array>
31
32      <string-array name="data_link">
33  <item>https://wetv.vip/id/play/a41kjezlclozg0i/y0040sv1hu
    9-EP01A%3A_Antares</item>
34      <item>https://wetv.vip/id/play/hm0734w9etht4vn-
    Harus%20Kawin/w0047tjoaoi-EP01A%3A%20Harus%20Kawin</item>
35  <item>https://wetv.vip/id/play/yweofusr9c2jpy2/z0041b49t1
    9-EP01A%3A_Kaget_Nikah</item>
36      <item>https://wetv.vip/id/play/k2cb8gl9t27lpbq-
    Jodoh%20atau%20Bukan/h0045cixsi1-
    EP01A%3A%20Jodoh%20atau%20Bukan</item>
37      <item>https://wetv.vip/id/play/ns0pvt3754wv3il-
    Kisah%20Untuk%20Geri/p0036f3bfbj-
    EP01%3A%20Kisah%20Untuk%20Geri</item>
38      <item>https://wetv.vip/id/play/3qoivvzv2pbi6wt-
    Mozachiko/w0046wfrhc5-EP01A%3A%20Mozachiko</item>
39      <item>https://wetv.vip/id/play/yfxtx6zpngrg25j-
    5%20Detik%20%26%20Rasa%20Rindu/b0047d2dh1h-
```

```
EP01A%3A%205%20Detik%20%26%20Rasa%20Rindu</item>
40      <item>https://wetv.vip/id/play/2s6dyeoedxdi6xc-
17%20Selamanya/o00429wd0s1-
EP01A%3A%2017%20Selamanya</item>
41      <item>https://wetv.vip/id/play/akwckof2lj9wjmn-
Teluk%20Alaska/i0041ztcg2t-
EP01A%3A%20Teluk%20Alaska</item>
42      <item>https://wetv.vip/id/play/ff3culwgrhb8q63-
Dua%20Wajah%20Arjuna/k0047b3g69u-
EP01A%3A%20Dua%20Wajah%20Arjuna</item>
43  </string-array>
44
45      <string-array name="data_plot">
46          <item>Series ini mengisahkan Zea, siswi baru yang
berusaha mengungkap kebenaran di balik kecelakaan
kakaknya yang melibatkan geng motor bernama Calderioz,
yang dipimpin oleh Antares.</item>
47          <item>Mengisahkan lima sahabat yang dipaksa
menikah oleh keluarga mereka. Serial ini menyoroti dilema
dan tekanan sosial yang mereka hadapi dalam menghadapi
pernikahan yang tidak diinginkan.</item>
48          <item>Lalita mengalami kecelakaan yang
menyebabkan kehilangan keperawanannya dan terpaksa
menikah dengan Andre, pemuda yang menolongnya. Pernikahan
ini dijalani tanpa cinta karena desakan orang tua.</item>
49          <item>Natalie, seorang wanita stylish, hidupnya
berubah saat ibunya menjodohkannya dengan anak partner
bisnisnya. Serial ini mengeksplorasi dilema antara cinta
dan perjodohan.</item>
50          <item>Setelah keluarganya bangkrut, Dinda, yang
dulunya populer, harus menghadapi realitas baru dan
```

```
    menjalin hubungan dengan Geri, mantan musuhnya, dalam
    sebuah kontrak palsu yang berubah menjadi cinta.</item>
51          <item>Moza, siswi culun, bertekad membuat Chiko,
    siswa populer, jatuh cinta padanya dalam 100 hari.</item>
52          <item>Via, seorang karyawan muda, mencari artis
    baru untuk duet dengan Drupadi. Gana pun ditunjuk sebagai
    teman duet, dan mereka merilis singel yang diadaptasi
    dari puisi karya Via.</item>
53          <item>Dawai, gadis polos yang disukai Putra,
    menyimpan rahasia besar terkait masa lalu mereka yang
    memengaruhi hubungan mereka.</item>
54          <item>Anastasia, gadis pendiam, bertemu kembali
    dengan Alister, teman masa kecilnya yang kini menjadi
    siswa populer.</item>
55          <item>Arjuna, pria dengan wajah biasa, menemukan
    spray misterius yang mengubah wajahnya menjadi tampan,
    mengubah kehidupannya secara drastis.</item>
56      </string-array>
57
58      <string-array name="data_year">
59          <item>30 Juli 2021</item>
60          <item>5 Januari 2024</item>
61          <item>23 Desember 2021</item>
62          <item>13 Januari 2023</item>
63          <item>5 Maret 2021</item>
64          <item>2 Juni 2023</item>
65          <item>8 September 2023</item>
66          <item>10 Maret 2022</item>
67          <item>5 November 2021</item>
68          <item>30 Oktober 2023</item>
69      </string-array>
```

```
70
71      <string-array name="data_cast">
72          <item>Angga Yunanda, Beby Tsabina, Irzan
    Faiq</item>
73          <item>Kevin Julio, Haico Van der Veken</item>
74          <item>Aurora Ribero, Kevin Julio, Steffi Zamora,
    Fero Walandouw</item>
75          <item>Megan Domani, Rayn Wijaya, Salshabilla
    Adriani</item>
76          <item>Angga Yunanda, Syifa Hadju </item>
77          <item>Rebecca Klopper, Junior Roberts, Yesaya
    Abraham</item>
78          <item>Prilly Latuconsina, Bryan Domani,
    Salshabilla Adriani</item>
79          <item>Syifa Hadju, Rizky Nazar, Steffi
    Zamora</item>
80          <item>Syifa Hadju, Devano Danendra, Kaneishia
    Yusuf</item>
81          <item>Yesaya Abraham, Beby Tsabina, Mikha
    Herman</item>
82      </string-array>
83
84      <string name="hello_blank_fragment">Hello blank
    fragment</string>
85  </resources>
```

Table 11. Source Code Jawaban Soal 1

File build.gradle.kts (:app)

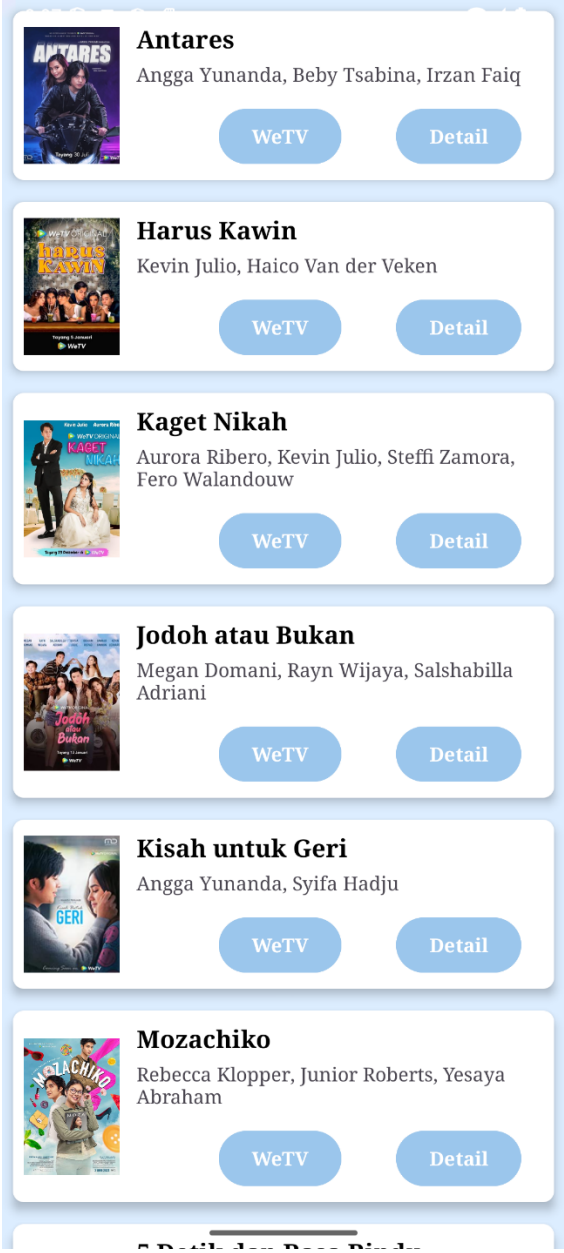
```
1   plugins {
2       id("com.android.application")
3       id("org.jetbrains.kotlin.android")
```

```
4       id("org.jetbrains.kotlin.plugin.parcelize")
5   }
6
7   android {
8       namespace = "com.example.recycler_view"
9       compileSdk = 35
10
11      defaultConfig {
12          applicationId = "com.example.recycler_view"
13          minSdk = 30
14          targetSdk = 35
15          versionCode = 1
16          versionName = "1.0"
17
18          testInstrumentationRunner =
    "androidx.test.runner.AndroidJUnitRunner"
19      }
20
21      buildTypes {
22          release {
23              isMinifyEnabled = false
24              proguardFiles(
25                  getDefaultProguardFile("proguard-android-
    optimize.txt"),
26                  "proguard-rules.pro"
27              )
28          }
29      }
30      compileOptions {
31          sourceCompatibility = JavaVersion.VERSION_11
32          targetCompatibility = JavaVersion.VERSION_11
```

```
33      }
34      kotlinOptions {
35          jvmTarget = "11"
36      }
37      buildFeatures {
38          viewBinding = true
39      }
40  }
41
42  dependencies {
43
44      implementation(libs.androidx.core.ktx)
45      implementation(libs.androidx.appcompat)
46      implementation(libs.material)
47      implementation(libs.androidx.activity)
48      implementation("androidx.recyclerview:recyclerview:1.3
    .1")
49      implementation
    ("androidx.constraintlayout:constraintlayout:2.1.4")
50      testImplementation(libs.junit)
51      androidTestImplementation(libs.androidx.junit)
52     androidTestImplementation(libs.androidx.espresso.core)
53  }
```

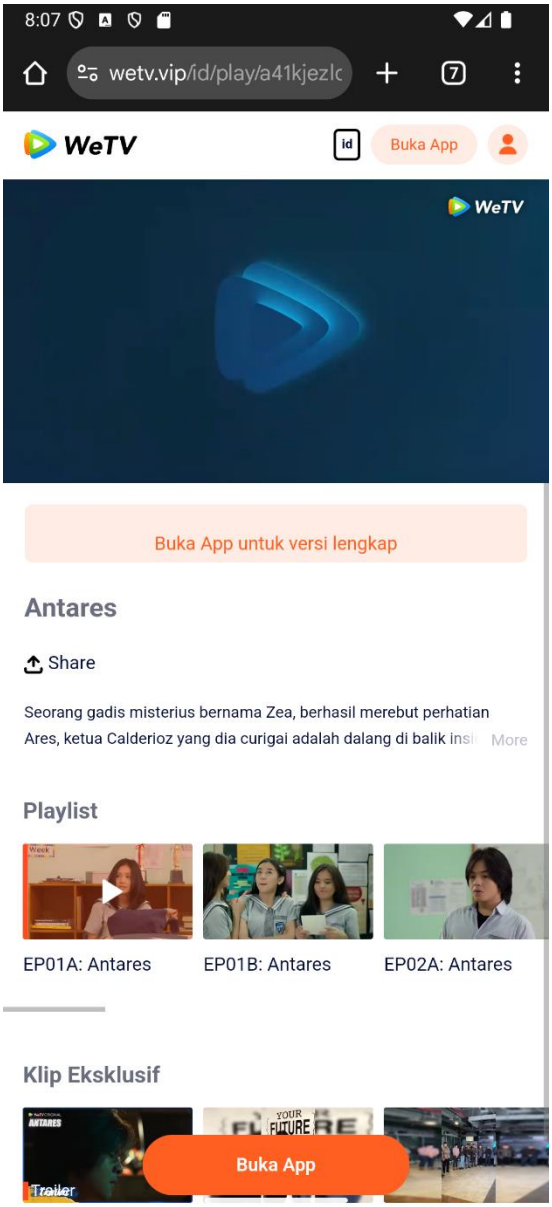
Table 12. Source Code Jawaban Soal 1

B. Output Program



Gambar 3. Screenshot Hasil Jawaban Soal 1

Gambar 4. Screenshot Hasil Jawaban Soal 1



Gambar 5. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

File MainActivity.kt

Kelas **`MainActivity`** mewarisi dari **`AppCompatActivity`**, yang merupakan superclass umum untuk aktivitas yang menggunakan fitur-fitur modern seperti **ActionBar**. Di dalam metode **`onCreate()`**, pertama-tama dipanggil **`super.onCreate(savedInstanceState)`** untuk memastikan bahwa logika

inisialisasi dari superclass berjalan sebagaimana mestinya. Selanjutnya, **`setContentView(R.layout.activity_main)`** digunakan untuk menentukan layout XML yang akan digunakan sebagai tampilan utama dari aktivitas ini, dalam hal ini adalah **`activity_main.xml`**.

Setelah layout diatur, kode melanjutkan dengan inisialisasi **`FragmentManager`** melalui **`supportFragmentManager`**. **`FragmentManager`** ini digunakan untuk mengelola transaksi fragment seperti penambahan, penggantian, atau penghapusan fragment. Kemudian, dibuat sebuah *instance* dari **`HomeFragment`**, yaitu fragment yang akan ditampilkan dalam activity. Baris **`val fragment = fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName)`** digunakan untuk memeriksa apakah fragment dengan tag tertentu yang berdasarkan nama kelas **`HomeFragment`** sudah ada atau belum dalam manajer fragment.

Jika fragment yang ditemukan bukan *instance* dari **`HomeFragment`** **`(fragment !is HomeFragment)`**, maka dilakukan transaksi untuk menambahkan fragment ke dalam container dengan ID **`R.id.frame_container`**. Metode **`beginTransaction()`** digunakan untuk memulai transaksi, kemudian **`add()`** menambahkan fragment ke dalam layout. Parameter **`HomeFragment::class.java.simpleName`** sebagai tag digunakan agar fragment ini bisa dikenali atau dicari kembali di kemudian hari. Terakhir, transaksi ini dikonfirmasi dan dijalankan dengan **`commit()`**.

File DetailFragment.kt

`DetailFragment` bertanggung jawab untuk menampilkan detail dari suatu series. **`DetailFragment`** mewarisi dari kelas **`Fragment`**, dan menggunakan *view binding* untuk mengakses elemen-elemen UI secara aman dan efisien tanpa perlu menggunakan **`findViewById`**.

Dalam fragment ini, terdapat properti **`_binding`** bertipe *nullable* **`FragmentDetailBinding`**, yang menyimpan referensi ke objek binding yang dihasilkan dari file layout **`fragment_detail.xml`**. Akses terhadap binding dilakukan

melalui properti **binding**, yang menggunakan operator **!!** untuk memastikan binding tidak null sehingga, hanya bisa diakses ketika view sudah dibuat.

Metode **onCreateView()** digunakan untuk meng-*inflate layout fragment* dan mengisi data ke dalam elemen UI. Pertama, objek **_binding** diinisialisasi menggunakan **FragmentDetailBinding.inflate(inflater, container, false)**, yang berarti *layout* dari fragment di-*inflate* dan dikaitkan dengan *binding*. Kemudian, data diambil dari *argument bundle* yang diberikan ke *fragment* melalui **arguments?.getString(...)** dan **arguments?.getInt(...)**. Nilai-nilai ini meliputi **EXTRA_TITLE**, **EXTRA_PHOTO, EXTRA_PLOT, EXTRA_YEAR**, dan **EXTRA_CAST**.

Setelah data diambil, fragment mengisi tampilan menggunakan binding, misalnya dengan **binding.tvName.text = title dan binding.imgItemPhoto.setImageResource(it)** untuk menampilkan gambar jika nilai **photo** tidak null. Metode ini mengembalikan **binding.root**, yaitu *root view* dari *layout* yang telah di-*inflate*. Untuk menghindari memory leak, **onDestroyView()** digunakan untuk menyetel `_binding` ke `null` saat view fragment dihancurkan.

File HomeFragment.kt

Fragment ini memanfaatkan fitur View Binding melalui **FragmentHomeBinding**, yang memberikan akses langsung dan aman ke elemen-elemen antarmuka pengguna tanpa harus menggunakan **findViewById()**. Dalam metode **onCreateView()**, layout fragment di-*inflate* dan dikembalikan sebagai root view. Kemudian, pada **onViewCreated()**, **RecyclerView** diatur menggunakan **LinearLayoutManager** untuk menampilkan daftar secara vertikal, dan dioptimalkan dengan **setHasFixedSize(true)** agar performanya lebih baik jika ukuran item tetap.

Sebuah adapter bernama **DramaAdapter** dibuat dan dikaitkan dengan **RecyclerView**. Adapter ini menerima dua aksi **callback** melalui **onWikiClick**, yang akan membuka tautan WeTV dari sebuah series menggunakan **Intent** dan browser

eksternal, serta **onDetailClick**, yang akan membuat instance **DetailFragment**, mengemas data series (seperti judul, gambar, alur cerita, tahun, dan pemeran) ke dalam **Bundle**, lalu menampilkan fragment detail tersebut dengan mengganti isi container menggunakan fragment transaction.

Data series diambil dari resource XML menggunakan metode **getDramaList()**. Data seperti judul, link, gambar, alur cerita, tahun rilis, dan pemeran diambil dari array resource, lalu setiap entri dikemas ke dalam objek **Series** dan dimasukkan ke dalam list yang kemudian diberikan ke adapter. **TypedArray** juga direcycle setelah digunakan untuk mencegah kebocoran memori. Terakhir, dalam **onDestroyView()**, binding disetel ke **null** untuk memastikan referensi ke tampilan dilepas ketika fragment dihancurkan, sehingga mencegah terjadinya memory leak.

File ListSeriesAdapter.kt

Kelas ini menampilkan daftar objek **Series** dalam bentuk daftar item. Adapter ini menerima tiga parameter utama dalam konstruktor sebuah **ArrayList<Series>** yang berisi data drama, sebuah lambda **onWikiClick** yang menangani aksi saat pengguna ingin membuka tautan WeTV, dan **onDetailClick** yang menangani aksi ketika pengguna ingin melihat detail drama.

Di dalam adapter terdapat kelas **ViewHolder**, yang bertanggung jawab untuk menampung dan menginisialisasi elemen-elemen UI dari setiap item, yaitu **ImageView** untuk **imgPhoto**, dua **TextView** untuk **tvTitle**, dan **tvPlot**, serta dua **Button** masing-masing untuk membuka **btnWiki** dan **btnDetail**. Layout setiap item dibuat dengan menggunakan file **item_list.xml** yang di-*inflate* di dalam metode **onCreateViewHolder()**. Metode ini mengembalikan objek **ViewHolder** yang siap digunakan.

Metode **getItemCount()** akan mengembalikan jumlah total item dalam daftar **listDrama**, yang menentukan seberapa banyak item yang akan ditampilkan di layar. Kemudian, pada **onBindViewHolder()**, adapter akan mengikat data ke elemen UI berdasarkan posisi item dalam daftar. Di sini, elemen-elemen seperti **title**, **photo**, dan **cast** dimasukkan ke tampilan masing-masing, dan tombol-tombol diatur dengan listener

yang memanggil fungsi callback yang telah didefinisikan. Misalnya, **btnWiki** akan membuka link WeTV menggunakan fungsi **onWikiClick**, sedangkan **btnDetail** akan memicu **onDetailClick** yang akan membawa data drama ke tampilan detail.

File Series.kt

Kelas ini menggunakan anotasi **@Parcelize** dan mengimplementasikan antarmuka **Parcelable**, yang memungkinkan objek **Series** untuk dikirim antar komponen Android (seperti antar **Activity** atau **Fragment**) melalui **Bundle**.

Setiap instance dari **Series** memiliki enam property yaitu, **title** bertipe **String** untuk menyimpan judul serial, **link** untuk menyimpan URL untuk mengarah ke web WeTV, **photo** bertipe **Int** yang merupakan resource ID dari drawable, **plot** yang berisi sinopsis cerita, **year** untuk tahun rilis, dan **cast** yang memuat daftar pemeran.

Dengan penggunaan **@Parcelize**, proses pembuatan parcel (objek yang bisa dipaketkan) menjadi otomatis, sehingga tidak perlu menuliskan kode boilerplate seperti **writeToParcel()** atau **describeContents()**.

File activity_main.xml

Kode XML tersebut merupakan layout utama dari **MainActivity** dalam aplikasi Android yang menggunakan **ConstraintLayout** sebagai root layout. Di dalamnya terdapat satu komponen penting, yaitu **FrameLayout** dengan ID **@+id/frame_container**. Elemen ini berfungsi sebagai **wadah** untuk menampung dan menampilkan **Fragment**, misalnya **HomeFragment** atau **DetailFragment**, yang dimuat secara dinamis melalui transaksi fragment dari kode Kotlin.

Tag **ConstraintLayout** menyediakan kemampuan untuk menempatkan dan mengatur posisi elemen-elemen UI secara fleksibel menggunakan constraint antar komponen atau ke parent layout. Namun dalam kasus ini, hanya ada satu elemen di dalamnya, yaitu **FrameLayout**. Meskipun begitu, karena ukuran **FrameLayout** diatur ke **match_parent** baik untuk lebar maupun tinggi, maka elemen ini akan mengisi seluruh ruang yang tersedia dalam layar.

Atribut **tools:context=".MainActivity"** digunakan oleh Android Studio saat preview layout di editor, untuk memberi tahu bahwa layout ini digunakan dalam **MainActivity**.

File fragment_detail.xml

Layout ini menggunakan **ScrollView** sebagai root. Di dalam **ScrollView**, terdapat **ConstraintLayout** sebagai container utama untuk menata elemen-elemen UI secara fleksibel. Elemen pertama adalah **ImageView** dengan ID **img_item_photo**, yang menampilkan gambar serial dengan ukuran tetap **(341dp x 458dp)** dan **scaleType centerCrop** agar gambar terisi penuh namun tetap proporsional. Gambar ini ditampilkan di bagian atas layout dan diberi margin atas **32dp**. Terdapat empat buah **TextView** yaitu, **tv_name** untuk menampilkan nama atau judul serial. ukuran teks **25sp, bold**, dan menggunakan font **serif** agar tampak khas seperti judul buku atau film, **tv_year** untuk menampilkan tahun rilis dengan label "Tahun:", **tv_cast** untuk menampilkan pemeran dari serial, diletakkan tepat di bawah tahun dan dipusatkan, **tv_plot** untuk menampilkan sinopsis atau alur cerita dari serial tersebut, juga dipusatkan dan diberi margin agar tampak rapi.

Semua elemen diposisikan menggunakan **ConstraintLayout**, yaitu dengan menentukan hubungan seperti **app:layout_constraintTop_toBottomOf, layout_constraintStart_toStartOf**, dan **layout_constraintEnd_toEndOf** agar konten tersusun secara vertikal dan simetris di tengah layar.

Atribut **android:background="@drawable/bg"** pada **ScrollView** menunjukkan bahwa latar belakang layar menggunakan gambar atau bentuk dari resource drawable dengan nama **bg.**

File fragment_home.xml

Kelas ini root layout-nya menggunakan **ConstraintLayout**, yang memberi fleksibilitas tinggi dalam menyusun elemen-elemen UI. Layout ini juga memiliki latar

belakang bergambar melalui atribut **android:background="@drawable/bg2",** yang memberikan estetika visual yang lebih menarik dibandingkan latar belakang polos.

Satu-satunya elemen yang dimuat dalam **ConstraintLayout** ini adalah **RecyclerView** dengan ID **rvDrama**. Komponen ini digunakan untuk menampilkan daftar series secara efisien, karena hanya memuat tampilan yang terlihat di layar dan mendaur ulang elemen lainnya. **RecyclerView** diatur agar mengisi seluruh area layar dengan menggunakan constraint ke setiap **Top, Bottom, Start, End**, serta lebar dan tinggi diset ke **0dp** yang berarti akan mengikuti constraint yang telah didefinisikan.

Layout ini bekerja bersama dengan **DramaAdapter** dan data dari **Series** untuk menampilkan daftar item yang interaktif, di mana pengguna bisa menekan tombol untuk membuka halaman detail atau tautan WeTV dari serial tersebut.

File item_list.xml

Root elemen dari layout ini adalah **CardView**, yang memberikan tampilan kartu dengan sudut melengkung dengan menggunakan **cardCornerRadius="8dp"** dan bayangan dengan menggunakan **cardElevation="4dp"**, menciptakan tampilan modern dan terangkat dari latar belakang.

Di dalam **CardView**, terdapat **ConstraintLayout** sebagai wadah utama untuk mengatur tata letak elemen UI seperti gambar, teks, dan tombol dengan posisi yang fleksibel dan presisi. Bagian kiri layout menampilkan **ImageView (@+id/img_item_photo)** berukuran **70x100dp** yang menampilkan gambar sampul serial dengan skala **centerCrop**, sehingga gambar akan memenuhi area tampilan secara proporsional.

Di samping gambar, terdapat dua **TextView:** satu untuk nama series dengan menggunakan **@+id/tv_item_name** dengan ukuran teks besar dan gaya tebal untuk menonjolkan judul, dan satu lagi di bawahnya yaitu **@+id/tv_item_plot** yang menampilkan informasi singkat seperti nama pemeran. Kedua teks ini menggunakan font **serif**, memberi nuansa klasik dan rapi. Di bagian bawah, terdapat dua tombol **Button** yang berfungsi sebagai aksi untuk pengguna. **btn_wiki** diberi label "WeTV" dan kemungkinan digunakan untuk membuka link eksternal yaitu WeTV. Tombol kedua yaitu

button_detail membuka halaman detail serial. Kedua tombol ini didesain dengan **backgroundTint** berwarna biru muda dengan kode warna **#9bc6ec**, sehingga tampak serasi.

File colors.xml

Dalam kelas ini terdapat beberapa warna yang didefiniskan untuk digunakan secara konsisten di seluruh aplikasi, yaitu:

- **black** – Diberi nilai **#FF000000**, yang merupakan warna hitam sepenuhnya.
- **white** – Diberi nilai **#FFFFFFFF**, warna putih dengan opasitas penuh.
- **blue** – Diberi nilai **#9bc6ec**, warna biru muda yang sering dipakai dalam elemen UI seperti latar belakang tombol (**backgroundTint**) agar terlihat segar dan menarik secara visual.

File strings.xml

Terdapat tiga string tunggal yaitu **app_name, btn_detail**, dan **btn_wiki** yang masing-masing digunakan sebagai nama aplikasi, label tombol detail, dan tombol WeTV. Selanjutnya, file ini berisi beberapa **string-array** yang digunakan untuk menampilkan data dinamis dalam daftar seperti **RecyclerView**. **data_name** berisi daftar judul drama, **data_plot** berisi sinopsis singkat tiap drama, **data_year** mencatat tanggal rilisnya, **data_cast** menampilkan nama-nama pemeran utama, dan **data_photo** untuk menampikan poster tiap series. Masing-masing elemen dalam array ini disusun secara paralel, artinya urutannya harus konsisten satu sama lain agar data bisa ditampilkan dengan benar.

File build.gradle.kts (:app)

Pada file **build.gradle.kts** (:app) yang ditampilkan merupakan berkas konfigurasi utama untuk aplikasi Android berbasis Kotlin yang menggunakan DSL (*Domain Specific Language*) Kotlin. File ini berperan dalam mendefinisikan proses build aplikasi, termasuk plugin yang digunakan, konfigurasi SDK Android, fitur build, serta daftar dependensi. Di bagian atas file, terdapat deklarasi tiga plugin utama:

com.android.application, **org.jetbrains.kotlin.android**, dan **org.jetbrains.kotlin.plugin.parcelize**. Plugin **com.android.application** digunakan untuk menyatakan bahwa modul ini adalah aplikasi Android. Plugin **kotlin.android** memungkinkan penggunaan bahasa Kotlin dalam pengembangan Android, sementara plugin **kotlin-parcelize** memungkinkan penggunaan anotasi **@Parcelize** untuk mempermudah implementasi **Parcelable**.

Blok **android** memuat konfigurasi inti proyek. Properti **namespace** diatur ke **"com.example.recycler_view"**, yang berfungsi sebagai ruang nama unik aplikasi. Versi **compileSdk**, **targetSdk**, dan **minSdk** masing-masing ditetapkan ke 35, 35, dan 30. Ini menandakan bahwa aplikasi dikembangkan menggunakan API level 35 dan dapat dijalankan pada perangkat Android dengan API level 30 ke atas (Android 11+). Bagian **defaultConfig** juga mendefinisikan **applicationId**, **versionCode**, dan **versionName**, yang diperlukan dalam distribusi dan pembaruan aplikasi di Google Play. **testInstrumentationRunner** disetel ke **androidx.test.runner.AndroidJUnitRunner** untuk keperluan pengujian instrumentasi.

Dalam blok **buildTypes**, terdapat konfigurasi **release** yang menonaktifkan proses **minifyEnabled**, artinya kode tidak akan dimampatkan saat proses rilis, dan **proguardFiles** menunjukkan penggunaan konfigurasi default ProGuard serta file **proguard-rules.pro** untuk aturan kustom. Kemudian, **compileOptions** dan **kotlinOptions** menetapkan bahwa proyek ini menggunakan Java 11 baik sebagai target maupun sumber kompatibilitas, serta mengarahkan Kotlin untuk menargetkan JVM versi 11.

Selanjutnya, fitur **viewBinding** diaktifkan melalui **buildFeatures**, yang memungkinkan pengaksesan langsung ke elemen UI tanpa menggunakan **findViewById**, sehingga meningkatkan keamanan dan efisiensi pengembangan.

Di bagian **dependencies**, proyek ini sudah menggunakan Version Catalog, namun hanya pada bagian library, bukan plugin. Hal ini terlihat dari penggunaan **libs.xxx** seperti **libs.androidx.core.ktx**, **libs.androidx.appcompat**, **libs.material**, **libs.androidx.activity**, serta dependensi eksplisit untuk

recyclerview dan **constraintlayout**. Untuk pengujian, digunakan **junit** untuk unit test, serta **androidx.junit** dan **espresso.core** untuk instrumented test. Penggunaan Version Catalog pada library memungkinkan pengelolaan versi yang lebih terpusat dan konsisten melalui file **libs.versions.toml**.

D. Tautan Git

https://github.com/biilaaa/Pemrograman-Mobile

SOAL 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 masih banyak digunakan karena stabilitas, fleksibilitas, dan dukungan ekosistemnya yang sangat matang. Sejak diperkenalkan sebagai pengganti ListView, RecyclerView telah menjadi standar de facto untuk menampilkan daftar data dalam Android UI berbasis View System. Meskipun memiliki kode yang lebih panjang dan memerlukan boilerplate seperti adapter, view holder, dan layout manager, RecyclerView menawarkan kontrol penuh atas perilaku tampilan daftar, seperti pengaturan item grid, drag & drop, swipe, animasi kompleks, dan integrasi dengan berbagai library eksternal. Banyak proyek lama juga masih menggunakan View System, sehingga RecyclerView tetap relevan demi kompatibilitas dan konsistensi.

Sebaliknya, LazyColumn adalah bagian dari Jetpack Compose, toolkit UI modern yang berbasis deklaratif dan memungkinkan penulisan UI yang lebih ringkas, reaktif, dan modular. LazyColumn mengurangi kebutuhan akan boilerplate karena tidak memerlukan adapter atau view holder, sehingga kode menjadi lebih bersih dan cepat ditulis. Namun, Compose masih tergolong baru, dan meskipun sudah stabil, adopsinya masih dalam transisi. Beberapa tim pengembang memilih tetap menggunakan RecyclerView karena mereka memiliki kode warisan (legacy code), kendala integrasi, atau belum sepenuhnya beralih ke Compose karena berbagai alasan seperti resource, kompatibilitas library pihak ketiga, atau preferensi tim.