LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Dina Izzati Elfadheya NIM. 2310817120001

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 : Dina Izzati Elfadheya
NIM : 2310817120001

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

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

A. Source Code

1. MainActivity.kt

```
1  package com.example.recyclerview
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,
   HomeFragment::class.java.simpleName)
18                 .commit()
19         }
20     }
21 }
```

Tabel 1 Source Code Jawaban MainActivity.kt

2. Drama.kt

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

Tabel 2 Source Code Jawaban Drama.kt

3. HomeFragment.kt

```
1  package com.example.recyclerview
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.recyclerview.databinding.FragmentHom
   eBinding
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         val dramaAdapter =
   DramaAdapter(getDramaList(),
31             onWikiClick = { link ->
32                 val uri = Uri.parse(link)
33 startActivity(Intent(Intent.ACTION_VIEW, uri))
34             },
35
```

```
            onDetailClick = { title, photo,
plot, year, cast ->
                val detailFragment =
DetailFragment().apply {
                    arguments = Bundle().apply {
                        putString("EXTRA_TITLE",
title)
                        putInt("EXTRA_PHOTO",
photo)
                        putString("EXTRA_PLOT",
plot)
                        putString("EXTRA_YEAR",
year)
                        putString("EXTRA_CAST",
cast)
                    }
                }
parentFragmentManager.beginTransaction()
.replace(R.id.frame_container, detailFragment)
                    .addToBackStack(null)
                    .commit()
            })
        binding.rvDrama.adapter = dramaAdapter
    }

    private fun getDramaList(): ArrayList<Drama>
{
        val dataTitle =
resources.getStringArray(R.array.data_name)
        val dataLink =
resources.getStringArray(R.array.data_link)
        val dataPhoto =
resources.obtainTypedArray(R.array.data_photo)
        val dataPlot =
resources.getStringArray(R.array.data_plot)
        val dataYear =
resources.getStringArray(R.array.data_year)
        val dataCast =
resources.getStringArray(R.array.data_cast)
        val listDrama = ArrayList<Drama>()
        for (i in dataTitle.indices) {
            val drama = Drama(
                dataTitle[i],
                dataLink[i],
                dataPhoto.getResourceId(i, -1),
                dataPlot[i],
                dataYear[i],
                dataCast[i]
            )
```

```
72              listDrama.add(drama)
73          }
74          dataPhoto.recycle()
75          return listDrama
76      }
77      override fun onDestroyView() {
78          super.onDestroyView()
79          _binding = null
80      }
81  }
```

Tabel 3 Source Code Jawaban HomeFragment.kt

4. DetailFragment

```
1   package com.example.recyclerview
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.bumptech.glide.Glide
9   import
    com.example.recyclerview.databinding.FragmentDet
    ailBinding
10
11  class DetailFragment : Fragment() {
12
13      private var _binding: FragmentDetailBinding?
    = null
14      private val binding get() = _binding!!
15
16      override fun onCreateView(
17          inflater: LayoutInflater, container:
    ViewGroup?,
18          savedInstanceState: Bundle?
19      ): View? {
20          _binding =
    FragmentDetailBinding.inflate(inflater,
    container, false)
21
22          val title =
    arguments?.getString("EXTRA_TITLE")
23          val photo =
    arguments?.getString("EXTRA_PHOTO")
24          val plot =
    arguments?.getString("EXTRA_PLOT")
```

```
25          val year =
    arguments?.getString("EXTRA_YEAR")
26          val cast =
    arguments?.getString("EXTRA_CAST")
27
28          binding.tvName.text = title
29          binding.tvPlot.text = plot
30          binding.tvYear.text = "Year: $year"
31          binding.tvCast.text = "Cast: $cast"
32          photo?.let {
33              Glide.with(requireContext())
34                  .load(it)
35                  .into(binding.imgItemPhoto)
36          }
37
38          binding.buttonKembali.setOnClickListener
39  {
40              parentFragmentManager.popBackStack()
41          }
42          return binding.root
43      }
44      override fun onDestroyView() {
45          super.onDestroyView()
46          _binding = null
47      }
48  }
```

Tabel 4 Source Code Jawaban DetailFragment.kt

5. DramaAdapter.kt

```
1   package com.example.recyclerview
2
3   import android.view.LayoutInflater
4   import android.view.View
5   import android.view.ViewGroup
6   import android.widget.Button
7   import android.widget.ImageView
8   import android.widget.TextView
9   import androidx.recyclerview.widget.RecyclerView
10  import com.bumptech.glide.Glide
11
12  class DramaAdapter(
13      private val listDrama: ArrayList<Drama>,
14      private val onWikiClick: (String) -> Unit,
15      private val onDetailClick: (String, String,
    String, String, String) -> Unit
```

```
16 ) :
17 RecyclerView.Adapter<DramaAdapter.ViewHolder>()
   {
18     class ViewHolder(itemView: View) :
   RecyclerView.ViewHolder(itemView) {
19         val imgPhoto: ImageView =
   itemView.findViewById(R.id.img_item_photo)
20         val tvTitle: TextView =
   itemView.findViewById(R.id.tv_item_name)
21         val tvPlot: TextView =
   itemView.findViewById(R.id.tv_item_plot)
22         val btnWiki: Button =
   itemView.findViewById(R.id.btn_wiki)
23         val btnDetail: Button =
   itemView.findViewById(R.id.button_detail)
24     }
25
26     override fun onCreateViewHolder(parent:
   ViewGroup, viewType: Int): ViewHolder {
27         val view: View =
   LayoutInflater.from(parent.context).inflate(R.la
   yout.item_list, parent, false)
28         return ViewHolder(view)
29     }
30
31     override fun getItemCount(): Int =
   listDrama.size
32
33     override fun onBindViewHolder(holder:
   ViewHolder, position: Int) {
34         val (title, link, photo, plot, year,
   cast) = listDrama[position]
35         holder.tvTitle.text = title
36         Glide.with(holder.itemView.context)
37             .load(photo)
38             .into(holder.imgPhoto)
39         holder.tvPlot.text = "Plot: $plot"
40         holder.btnWiki.setOnClickListener {
   onWikiClick(link) }
42         holder.btnDetail.setOnClickListener {
   onDetailClick(title, photo, plot, year, cast) }
43     }
44 }
```

Tabel 5 Source Code Jawaban DramaAdapter.kt

6. Activity_main.xml

```
1  <androidx.constraintlayout.widget.ConstraintLayo
   ut
2  xmlns:android="http://schemas.android.com/apk/re
   s/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      tools:context=".MainActivity">
8
9      <FrameLayout
10         android:id="@+id/frame_container"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent" />
13
14 </androidx.constraintlayout.widget.ConstraintLay
   out>
```

Tabel 6 Source Code Jawaban Activity_Main.xml

7. fragment_home.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayo
   ut
3  xmlns:android="http://schemas.android.com/apk/re
   s/android"
4  xmlns:app="http://schemas.android.com/apk/res-
   auto"
5  xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".HomeFragment">
9
10     <androidx.recyclerview.widget.RecyclerView
11         android:id="@+id/rv_Drama"
12         android:layout_width="0dp"
13         android:layout_height="0dp"
14         android:layout_margin="15dp"
15
16 app:layout_constraintTop_toTopOf="parent"

17 app:layout_constraintBottom_toBottomOf="parent"

18 app:layout_constraintStart_toStartOf="parent"
```

```
19
20  app:layout_constraintEnd_toEndOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLay
    out>
```

Tabel 7 Source Code Jawaban fragment_home.kt

8. fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayo
    ut
3   xmlns:android="http://schemas.android.com/apk/re
    s/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   android:background="@drawable/background_image"
9       tools:context=".DetailFragment">
10
11      <ImageView
12          android:id="@+id/img_item_photo"
13          android:layout_width="341dp"
14          android:layout_height="458dp"
15          android:layout_marginTop="32dp"
16          android:scaleType="centerCrop"
17  app:layout_constraintEnd_toEndOf="parent"
18  app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent"
20          tools:src="@drawable/truebeauty" />
21
22      <TextView
23          android:id="@+id/tv_name"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:textStyle="bold"
27          android:textSize="20sp"
28          android:layout_marginTop="16dp"
29  app:layout_constraintTop_toBottomOf="@id/img_ite
    m_photo"
30  app:layout_constraintStart_toStartOf="parent"
31         app:layout_constraintEnd_toEndOf="parent"
32          tools:text="Drama Title" />
33
34      <TextView
35          android:id="@+id/tv_year"
36          android:layout_width="wrap_content"
```

```
37          android:layout_height="wrap_content"
38          android:text="Year: "
39          android:textSize="16sp"
40          android:layout_marginTop="8dp"
41  app:layout_constraintTop_toBottomOf="@id/tv_name
    "
42  app:layout_constraintStart_toStartOf="parent"
43         app:layout_constraintEnd_toEndOf="parent"
    />
44
45      <TextView
46          android:id="@+id/tv_cast"
47          android:layout_width="0dp"
48          android:layout_height="wrap_content"
49          android:text="Cast: "
50          android:textSize="16sp"
51          android:layout_marginTop="8dp"
52          android:layout_marginHorizontal="16dp"
53          android:textAlignment="center"
54  app:layout_constraintTop_toBottomOf="@id/tv_year
    "
55  app:layout_constraintStart_toStartOf="parent"
56  app:layout_constraintEnd_toEndOf="parent" />
57
58      <TextView
59          android:id="@+id/tv_plot"
60          android:layout_width="0dp"
61          android:layout_height="wrap_content"
62          android:layout_margin="16dp"
63          android:textAlignment="center"
64          android:textSize="16sp"
65          android:text="Plot: "
66  app:layout_constraintTop_toBottomOf="@id/tv_cast
    "
67  app:layout_constraintStart_toStartOf="parent"
68         app:layout_constraintEnd_toEndOf="parent"
    />
69
70  </androidx.constraintlayout.widget.ConstraintLay
    out>
```

Tabel 8 Source Code Jawaban fragment_detail.kt

9. item_list.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
3
```

```
4   xmlns:android="http://schemas.android.com/apk/re
    s/android"
5    xmlns:app="http://schemas.android.com/apk/res-
6   auto"
7       android:layout_width="match_parent"
8       android:layout_height="wrap_content"
9     xmlns:tools="http://schemas.android.com/tools"
10      android:layout_margin="8dp"
11      app:cardCornerRadius="8dp"
12      app:cardElevation="4dp">

13  <androidx.constraintlayout.widget.ConstraintLayo
14  ut
15          android:layout_width="match_parent"
16          android:layout_height="wrap_content"
17          android:padding="8dp">
18
19          <ImageView
20              android:id="@+id/img_item_photo"
21              android:layout_width="70dp"
22              android:layout_height="100dp"
23              android:scaleType="centerCrop"
24  app:layout_constraintStart_toStartOf="parent"
25  app:layout_constraintTop_toTopOf="parent"
26  app:layout_constraintBottom_toBottomOf="parent"
27              tools:src="@drawable/truebeauty" />
28
29          <TextView
30              android:id="@+id/tv_item_name"
31              android:layout_width="0dp"
32              android:layout_height="wrap_content"
33              android:textSize="16sp"
                android:textStyle="bold"
34  app:layout_constraintStart_toEndOf="@id/img_item
35  _photo"
36  app:layout_constraintTop_toTopOf="parent"
37  app:layout_constraintEnd_toEndOf="parent"
38              android:layout_marginStart="12dp"
39              tools:text="True Beauty" />
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"
                android:textSize="13sp"
46  app:layout_constraintTop_toBottomOf="@id/tv_item
    _name"
```

```
47  app:layout_constraintStart_toStartOf="@id/tv_ite
48  m_name"
49  app:layout_constraintEnd_toEndOf="parent"
50              tools:text="Plot: " />
51
52          <Button
53              android:id="@+id/btn_wiki"
54              android:layout_width="wrap_content"
55              android:layout_height="wrap_content"
56              android:layout_marginStart="24dp"
57              android:layout_marginTop="8dp"
                android:backgroundTint="#ffc5d3"
                android:text="IMDB"
58
    app:layout_constraintStart_toStartOf="@id/tv_ite
59  m_plot"
    app:layout_constraintTop_toBottomOf="@id/tv_item
60  _plot" />
61
62          <Button
63              android:id="@+id/button_detail"
64              android:layout_width="wrap_content"
65              android:layout_height="wrap_content"
66              android:layout_marginStart="8dp"
67              android:backgroundTint="#ffc5d3"
68              android:text="Detail"
    app:layout_constraintStart_toEndOf="@id/btn_wiki
69  "
    app:layout_constraintTop_toTopOf="@id/btn_wiki"
70  />
71
72
    </androidx.constraintlayout.widget.ConstraintLay
73  out>
    </androidx.cardview.widget.CardView>
```

Tabel 9 Source Code Jawaban item_list.kt

10. string.xml

```
1   <resources>
2       <string name="app_name">modul3</string>
3       <string name="btn_detail">Detail</string>
4       <string name="btn_wiki">Wiki</string>
5
6       <string-array name="data_name">
7           <item>True Beauty</item>
8           <item>Business Proposal</item>
9           <item>King the Land</item>
```

```
        <item>Welcome to Samdalri</item>
        <item>Love Next Door</item>
        <item>Family by Choice</item>
        <item>Lovely Runner</item>
        <item>Melo Movie</item>
    </string-array>

    <string-array name="data_photo">
        <item>@drawable/truebeauty</item>
        <item>@drawable/businessproposal</item>
        <item>@drawable/kingtheland</item>
        <item>@drawable/welcometosamdalri</item>
        <item>@drawable/lovenextdoor</item>
        <item>@drawable/familybychoice</item>
        <item>@drawable/lovelyrunner</item>
        <item>@drawable/melomovie</item>
    </string-array>

<string-array name="data_link">
<item>https://www.netflix.com/id/title/81410834</item>
    <item>https://www.netflix.com/id-en/title/81509440</item>
<item>https://www.netflix.com/id/title/81671440</item>
<item>https://www.netflix.com/id/title/81697983</item>
    <item>https://www.netflix.com/id-en/title/81739044</item>

<item>https://www.viu.com/ott/id/id/vod/2435994/Family-By-Choice</item>
    <item>https://www.netflix.com/id-en/title/81914565</item>
    <item>https://www.netflix.com/id-en/title/81720593</item>
</string-array>

    <string-array name="data_plot">
        <item>True Beauty is a high school drama about a girl who becomes a beauty through makeup and faces challenges when her secret is revealed.</item>
        <item>Business Proposal tells the story of a woman who participates in a business proposal pretending to be her friend, only to find out the man is her company s CEO</item>
        <item>King the Land is about a CEO who tries to open a new hotel and a woman who works
```

```
    there while struggling with her own love
    story.</item>
43          <item>Welcome to Samdalri is about a
    group of individuals who meet in a small village
    and form bonds through humorous and touching
    moments.</item>
44          <item>Love Next Door is a romantic drama
    revolving around a couple living next door and
    their developing relationship.</item>
45          <item>Family by Choice is a heartwarming
    drama about a group of people who form a family
    despite not being related by blood.</item>
46          <item>Lovely Runner follows a runner who
    is trying to make it in the world of competitive
    sports while balancing her personal life.</item>
47          <item>Melo Movie is a romantic comedy
    about a woman trying to find love while
    struggling with her career in the film
    industry.</item>
48      </string-array>
49
50      <string-array name="data_year">
51          <item>2020</item>
52          <item>2022</item>
53          <item>2023</item>
54          <item>2023</item>
55          <item>2024</item>
56          <item>2024</item>
57          <item>2024</item>
58          <item>2025</item>
59      </string-array>
60
61      <string-array name="data_cast">
62          <item>Moon Ga-young, Cha Eun-woo, Hwang
    In-yeop</item>
63          <item>Kim Se-jeong, Ahn Hyo-seop, Kim
    Min Kyu, Seol In Ah</item>
64          <item>YoonA, Lee Jun-ho</item>
65          <item>Ji Chang-wook, Shin Hye-sun</item>
66          <item>Jung Hae In, Jung So Min, Kim Ji
    Eun, Yun Ji On</item>
67          <item>Hwang In Yeop, Jung Chae Yeon, Bae
    Hyun Sung</item>
68          <item>Byeon Woo-seok, Kim Hye-
    yoon</item>
69          <item>Choi Woo Shik, Park Bo Young, Lee
    Jun Young, Jeon So Nee</item>
70      </string-array>
71
```

```
72
73
74      <string name="hello_blank_fragment">Hello
    blank fragment</string>
75  </resources>
```

Tabel 10 Source Code Jawaban string.kt

11. build.gradle.kts (module:app)

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

```
34      }
35      kotlinOptions {
36          jvmTarget = "11"
37      }
38      buildFeatures {
39          viewBinding = true
40      }
41  }
42
43  dependencies {
44
45      implementation(libs.androidx.core.ktx)
46      implementation(libs.androidx.appcompat)
47      implementation(libs.material)
48      implementation(libs.androidx.activity)
49  implementation("androidx.recyclerview:recyclervi
    ew:1.3.1")
50      implementation
    ("androidx.constraintlayout:constraintlayout:2.1
    .4")
51      testImplementation(libs.junit)
52  androidTestImplementation(libs.androidx.junit)
53  androidTestImplementation(libs.androidx.espresso
    .core)
54  implementation("com.github.bumptech.glide:glide:
    4.15.1")
55  kapt("com.github.bumptech.glide:compiler:4.15.1"
    )
56  }
57
58  fun kapt(s: String) {
59
60  }
```

Tabel 11 Source Code Jawaban build. gradle.kts

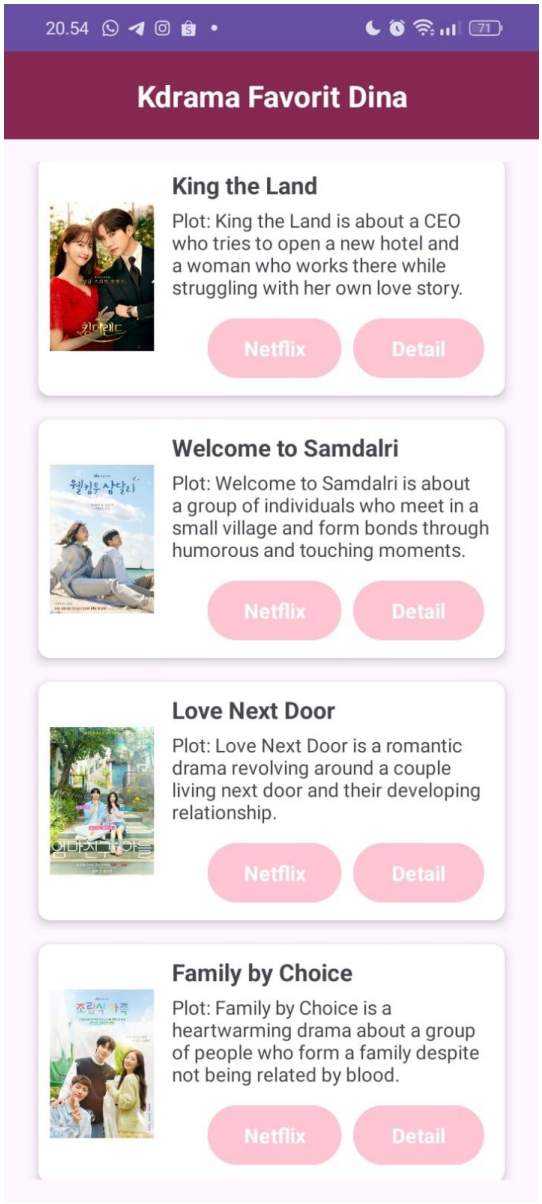
12. AndroidManifest.kt

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <manifest
3   xmlns:android="http://schemas.android.com/apk/re
4   s/android"
5
6   xmlns:tools="http://schemas.android.com/tools">
7
8       <application
9           android:allowBackup="true"
10
11  android:dataExtractionRules="@xml/data_extractio
12  n_rules"
```

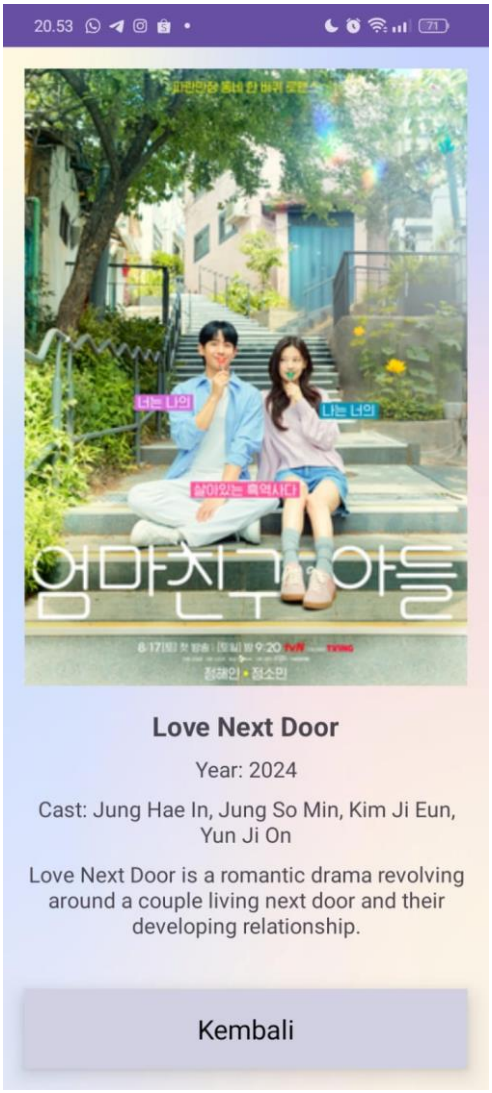
```
13
14  android:fullBackupContent="@xml/backup_rules"
15          android:icon="@mipmap/ic_launcher"
16          android:label="@string/app_name"
17
18  android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
19
20  android:theme="@style/Theme.RecyclerView"
21        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

    </manifest>
```

Tabel 12 Source Code Jawaban AndoidManifest.kt

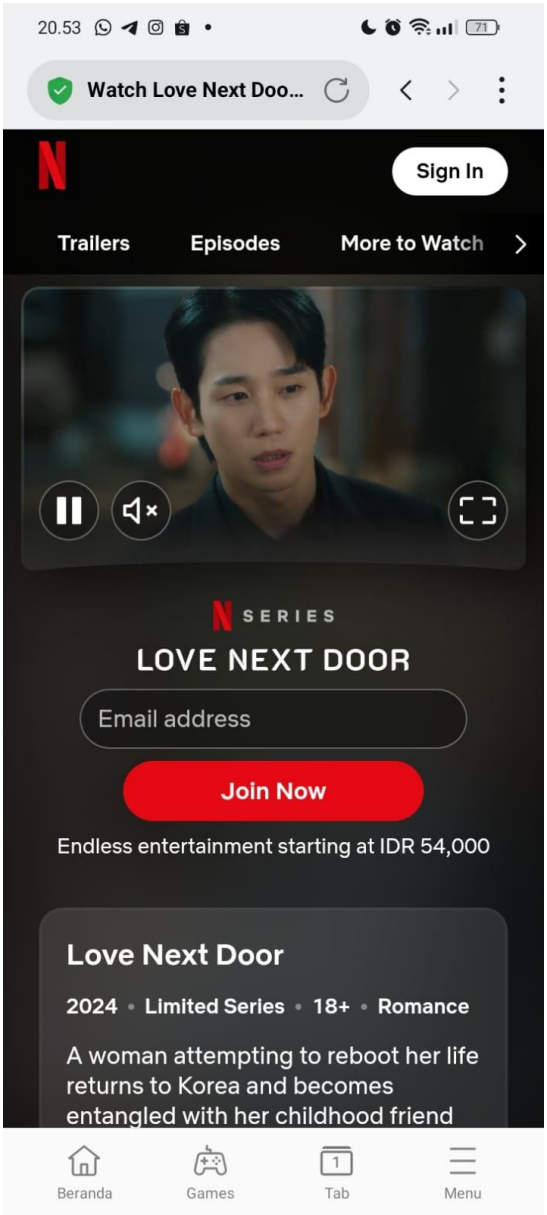
B. Output Program



Gambar 1 Halaman Home



Gambar 2 Halaman Detail



Gambar 3 Link Netflix

C. Pembahasan

1. MainActivity.kt

Pada baris 1 `package com.example.recyclerview` menunjukkan bahwa file Kotlin ini berada di dalam paket bernama recyclerview di bawah domain com.example. Baris 3–4 adalah bagian impor yang memungkinkan kelas ini menggunakan fitur seperti Bundle dan `AppCompatActivity` dari Android SDK. Baris 6 mendefinisikan kelas `MainActivity` yang merupakan titik masuk utama ketika aplikasi dijalankan. Baris 7 menjalankan fungsi `onCreate()` yang otomatis dipanggil saat activity dibuat. Baris 9 memanggil `setContentView(R.layout.activity_main)` untuk menampilkan layout `activity_main.xml` sebagai antarmuka pengguna. Baris 11 membuat objek `fragmentManager` untuk mengelola fragmen dalam activity ini. Baris 12 membuat instance dari kelas `HomeFragment`. Baris 13–14 mengecek apakah fragmen dengan tag yang sama sudah ditambahkan sebelumnya, dan jika belum, maka fragmen baru akan ditambahkan ke dalam `frame_container` dengan memanggil `beginTransaction()`, lalu commit() untuk menyimpan perubahan.

2. Drama.kt

Pada bagian awal, kode `package com.example.recyclerview` digunakan untuk mengelompokkan file ini ke dalam paket recyclerview agar lebih terorganisir dalam struktur proyek. Baris impor `import android.os.Parcelable` dan `import kotlinx.parcelize.Parcelize` memungkinkan kelas `Drama` dapat dikirim antar komponen Android dengan cara efisien menggunakan parcel. Anotasi `@Parcelize` digunakan supaya proses implementasi Parcelable tidak perlu dilakukan secara manual, cukup dengan menandai data class tersebut. Kelas `Drama` bertipe `data class` yang menyimpan informasi seperti `title`, `link`, `photo`, `plot`, `year`, dan `cast` yang masing-masing bertipe `String` Kelas ini mengimplementasikan

`Parcelable`, sehingga objeknya dapat dipaketkan dan ditransfer antar Activity atau Fragment.

3. HomeFragment.kt

Pada baris 1 `package com.example.recyclerview` menunjukkan bahwa file Kotlin ini berada dalam paket bernama `recyclerview` di bawah domain `com.example`. Baris 3-11 `import` yang mengikuti memungkinkan penggunaan berbagai komponen Android seperti `Intent`, `Uri`, dan `Fragment`. Baris 13 `class HomeFragment : Fragment()` mendeklarasikan bahwa `HomeFragment` adalah turunan dari kelas `Fragment`. Baris 15-16 `private var _binding: FragmentHomeBinding? = null` digunakan untuk menginisialisasi view binding guna mengakses elemen UI tanpa perlu `findViewById`. Baris 1851 `onCreateView` meng-inflate layout dari `FragmentHomeBinding` dan mengembalikannya sebagai tampilan utama dari fragmen. Di dalam `onViewCreated`, RecyclerView disiapkan dengan `LinearLayoutManager` untuk menampilkan daftar secara vertikal dan ukuran tetap. Sebuah adapter `DramaAdapter` diinisialisasi dengan data dari `getDramaList()` serta dua lambda untuk aksi klik: satu membuka tautan Netflix menggunakan `Intent.ACTION_VIEW`, dan satu lagi mengganti fragmen saat tombol detail diklik dengan meneruskan data ke `DetailFragment` melalui `Bundle`. Fungsi pada baris 53-73 `getDramaList()` mengambil data dari resources (seperti judul, foto, plot, tahun, dan pemeran), lalu memasukkannya ke dalam list bertipe `Drama`. Terakhir, baris 75-78 `onDestroyView` bertugas untuk menghapus binding agar tidak terjadi memory leak saat fragment dihancurkan.

4. DetailFragment,kt

Pada baris 1 `package com.example.recyclerview` menunjukkan bahwa file Kotlin ini berada dalam paket bernama `recyclerview` di bawah domain `com.example`. Baris 3-8 `import` yang mengikuti memungkinkan penggunaan berbagai komponen Android seperti `Bundle`, `Fragment`, dan `ViewBinding`. Baris 10 `class DetailFragment : Fragment()` mendeklarasikan bahwa `DetailFragment` merupakan subclass dari `Fragment`. Baris 12-13 `private var _binding: FragmentDetailBinding? = null` digunakan untuk mengelola binding agar bisa mengakses elemen UI dari `fragment_detail.xml` secara efisien. Baris 15-34 di dalam `onCreateView`, layout difungsikan melalui `FragmentDetailBinding.inflate`, lalu data yang dikirim melalui `arguments` diambil menggunakan kunci seperti `EXTRA_TITLE`, `EXTRA_PHOTO`, dan lainnya. Nilai-nilai tersebut kemudian diatur ke elemen UI yang sesuai seperti `tvName`, `tvPlot`, `tvYear`, `tvCast`, dan gambar `imgItemPhoto` apabila tersedia. Fungsi ini mengembalikan tampilan dari root binding. Baris 32-26 digunakan library Glide. Glide adalah library pihak ketiga yang efisien untuk menangani pemuatan gambar dari internet. Pada baris yang menggunakan `Glide.with(requireContext()).load(it).into(binding.imgItemPhoto)`, Glide digunakan untuk memuat gambar dari URL yang diperoleh dari variabel `photo`, kemudian menampilkannya ke dalam elemen `ImageView` bernama `imgItemPhoto`. Terakhir, pada baris 38-48 fungsi `onDestroyView` memastikan binding dihapus untuk mencegah kebocoran memori saat fragment dihancurkan.

5. DramaAdapter

Pada baris 1 `package com.example.recyclerview` menunjukkan bahwa file Kotlin ini berada dalam paket bernama `recyclerview` di bawah domain `com.example`. Baris 3-9 `import` yang mengikuti

memungkinkan penggunaan komponen Android seperti `RecyclerView`, `View`, `ImageView`, dan `TextView`. Baris 11 `class DramaAdapter` mendefinisikan adapter untuk RecyclerView dengan parameter daftar data bertipe `Drama`, lambda `onWikiClick` untuk membuka tautan, dan `onDetailClick` untuk membuka detail drama. Pada baris 17-23 di dalam kelas ini terdapat `ViewHolder` yang merepresentasikan satu item tampilan dan menghubungkan komponen UI seperti gambar, judul, plot, dan dua tombol aksi dari layout `item_list.xml`. Baris 25 Fungsi `onCreateViewHolder` bertugas untuk meng-inflate layout item dan mengembalikannya sebagai objek `ViewHolder`. Baris 30 Fungsi `getItemCount` mengembalikan jumlah item dalam daftar data. Pada baris 33-43 `onBindViewHolder`, data drama diambil berdasarkan posisi lalu diatur ke elemen tampilan masing-masing, dan dua tombol diberi listener sesuai fungsi yang diterima melalui konstruktor adapter. Dalam baris ini terdapat glide `Glide.with(holder.itemView.context).load(photo).into(holder.imgPhoto)` menunjukkan bahwa Glide akan menggunakan context dari item view saat ini, memuat gambar dari URL yang diberikan oleh variabel `photo`, lalu menampilkannya ke dalam `ImageView` yang memiliki ID `img_item_photo`.

6. Activity_Main.xml

Baris 1 `<androidx.constraintlayout.widget.ConstraintLayout>` adalah layout utama yang digunakan sebagai wadah seluruh elemen UI di halaman ini. Baris 2-4 adalah deklarasi namespace XML yang memungkinkan penggunaan atribut Android, app (untuk ConstraintLayout), dan tools. Baris 5-7 mendefinisikan ukuran layout agar memenuhi seluruh layar serta menetapkan konteks tools untuk keperluan preview di Android Studio. Baris 9-12 `<FrameLayout>` dengan id `frame_container` digunakan sebagai kontainer fragmen yang akan diganti-ganti secara

dinamis, misalnya saat navigasi antar halaman. Terakhir, baris 14 menutup tag ConstraintLayout sebagai akhir dari struktur layout ini.

7. fragment_home.xml

Baris 1 `<androidx.constraintlayout.widget.ConstraintLayout>` digunakan sebagai layout utama yang memungkinkan penempatan elemen UI secara fleksibel dengan constraint. Baris 2-4 adalah deklarasi namespace XML standar agar atribut seperti `android:`, `app:`, dan `tools:` dapat digunakan dengan benar. Baris 5-7 mengatur layout agar mengisi seluruh layar dan menyetel konteks preview di Android Studio ke `HomeFragment`. Baris 9-22 `<TextView>` dengan id `tvHeader` menampilkan teks dari resource `@string/header` sebagai judul halaman dengan ukuran 20sp, teks putih tebal, latar belakang ungu gelap, dan posisi berada di bagian atas layout serta disejajarkan secara horizontal di tengah. Baris 24-32 `<androidx.recyclerview.widget.RecyclerView>` dengan id `rv_Drama` digunakan untuk menampilkan daftar drama dalam bentuk list yang bisa discroll secara vertikal, ditempatkan tepat di bawah `tvHeader` dengan margin di sekelilingnya, dan memenuhi sisa ruang layar. Terakhir, baris 33 menutup tag ConstraintLayout sebagai akhir dari struktur layout ini.

8. fragment_detail.xml

Baris 1 `<androidx.constraintlayout.widget.ConstraintLayout>` digunakan sebagai layout utama yang memungkinkan penempatan elemen UI secara fleksibel dengan constraint. Baris 3-5 adalah deklarasi namespace XML standar agar atribut seperti android:, app:, dan tools: dapat digunakan dengan benar. Baris 6-9 mengatur layout agar mengisi seluruh layar, menambahkan latar belakang berupa gambar drawable, dan menyetel konteks preview ke DetailFragment. Baris 11-20 `<ImageView>` dengan id `img_item_photo` menampilkan gambar poster drama dengan ukuran tetap, skala centerCrop, serta diposisikan di bagian atas layout dengan margin atas

32dp dan posisi horizontal di tengah. Baris 22-32 `<TextView>` dengan id `tv_name` menampilkan judul drama dengan teks tebal berukuran 20sp, ditempatkan tepat di bawah gambar dan disejajarkan di tengah. Baris 34-43 `<TextView>` dengan id `tv_year` digunakan untuk menampilkan tahun rilis drama, dengan ukuran teks 16sp dan margin atas 8dp, diletakkan di bawah judul dan diposisikan secara tengah. Baris 45-56 `<TextView>` dengan id `tv_cast` menampilkan informasi pemeran dengan ukuran teks 16sp dan alignment tengah, diletakkan di bawah tahun dengan margin horizontal 16dp agar isi tidak menempel ke tepi layar. Baris 58-68 `<TextView>` dengan id `tv_plot` menampilkan sinopsis atau plot drama, ditempatkan di bawah informasi cast, dengan margin 16dp dan alignment tengah agar rapi dibaca. Terakhir, baris 70 menutup tag ConstraintLayout sebagai akhir dari struktur layout ini.

9. item_list.xml

Baris 1 `<androidx.cardview.widget.CardView>` digunakan sebagai wadah utama dengan sudut membulat dan efek elevasi agar tampilan setiap item terlihat seperti kartu. Baris 3-5 adalah deklarasi namespace XML standar yang memungkinkan penggunaan atribut seperti `android:`, `app:`, dan `tools:`. Baris 5-7 mengatur lebar kartu agar memenuhi parent, tinggi menyesuaikan isi, dengan margin 8dp serta pengaturan radius sudut 8dp dan elevasi 4dp agar tampak sedikit melayang. Baris 12-15 `<androidx.constraintlayout.widget.ConstraintLayout>` digunakan sebagai layout dalam kartu untuk menyusun elemen-elemen UI secara terstruktur. Baris 17-25 `<ImageView>` dengan id `img_item_photo` digunakan untuk menampilkan gambar poster drama berukuran 70x100dp dengan skala `centerCrop`, diposisikan di sisi kiri dan vertikal tengah dalam kartu. Baris 27-37 `<TextView>` dengan id `tv_item_name` menampilkan judul drama dengan teks tebal berukuran 16sp, diletakkan di samping gambar dengan margin awal 12dp dan disejajarkan ke kanan. Baris 39-48 `<TextView>` dengan id `tv_item_plot` menampilkan deskripsi singkat atau

plot drama di bawah judul, dengan ukuran teks lebih kecil 13sp dan sedikit margin atas. Baris 52-59 `<Button>` dengan id `btn_wiki` berfungsi untuk membuka tautan Netflix, diberi teks “Netflix”, latar belakang berwarna pink muda, dan diletakkan di bawah plot dengan margin awal 24dp. Baris 61-69 `<Button>` dengan id `button_detail` digunakan untuk menampilkan detail drama, posisinya tepat di samping kanan tombol Netflix dan tetap sejajar secara vertikal. Terakhir, baris 71 menutup tag ConstraintLayout dan baris 72 menutup tag CardView sebagai akhir dari struktur layout ini

10. string.xml

Baris 1 `<resources>` merupakan elemen root pada file XML ini yang berisi semua data string dan array yang digunakan dalam aplikasi. Baris 2-4 mendefinisikan string statis seperti nama aplikasi dan label pada tombol untuk detail serta wiki. Baris 6-15 `<string-array name="data_name">` menyimpan daftar judul drama Korea yang akan ditampilkan dalam RecyclerView. Baris 17-26 `<string-array name="data_photo">` menyimpan daftar URL gambar dari internet yang digunakan sebagai foto untuk setiap judul drama. Karena aplikasi menggunakan library Glide untuk memuat gambar. Baris 28-37 `<string-array name="data_link">` berisi link menuju halaman resmi masing-masing drama, kebanyakan dari Netflix dan satu dari Viu. Baris 39-48 `<string-array name="data_plot">` berisi deskripsi singkat atau sinopsis dari setiap drama, menjelaskan tema dan alur utama cerita. Baris 50-59 `<string-array name="data_year">` menyimpan tahun rilis masing-masing drama agar bisa ditampilkan pada tampilan detail. Baris 61-70 `<string-array name="data_cast">` memuat daftar pemeran utama dari masing-masing drama. Baris 73 dan 74 menambahkan string statis untuk keperluan default fragment dan judul header halaman. Terakhir, baris 75 menutup tag `<resources>` sebagai akhir dari seluruh deklarasi string dan array yang ada di file ini.

11. Build.gradle.kts

Pada gradle ini saya menambahkan baris 3 yaitu `id("kotlin-parcelize")` digunakan untuk mengaktifkan plugin Kotlin Parcelize yang memungkinkan penggunaan anotasi `@Parcelize` pada kelas data agar objeknya bisa dengan mudah dikirim antar komponen Android seperti Activity atau Fragment melalui `Intent` atau `Bundle`. Baris 4 Plugin `id("org.jetbrains.kotlin.kapt")` diaktifkan agar proyek dapat menggunakan KAPT (Kotlin Annotation Processing Tool), yang diperlukan ketika menggunakan library yang mengandalkan annotation processor Glide. Lalu pada dependencies menambahkan `implementation("androidx.recyclerview:recyclerview:1.3.1")` yang digunakan untuk menyertakan library RecyclerView versi 1.3.1 unyuk menampilkan daftar item secara efisien dan fleksibel dalam bentuk list atau grid yang dapat discroll. Dan `implementation("androidx.constraintlayout:constraintlayout:2.1.4")` digunakan untuk memasukkan library ConstraintLayout versi 2.1.4 yang memberikan kontrol layout yang lebih fleksibel dan efisien dengan sistem constraint antar elemen UI. Baris 54 `implementation("com.github.bumptech.glide:glide:4.15.1")` berfungsi untuk mengimpor library utama Glide versi 4.15.1, yang memungkinkan aplikasi Android memuat gambar dari berbagai sumber, termasuk URL di internet. baris `kapt("com.github.bumptech.glide:compiler:4.15.1")` digunakan untuk mengaktifkan dukungan annotation processor yang disediakan oleh Glide.

12. AndroidManifest.xml

Pada file ini, saya menambahkan `<intent-filter>` di dalam deklarasi `<activity android:name=".MainActivity">` yang berfungsi untuk menetapkan `MainActivity` sebagai titik masuk utama aplikasi. Baris `<action android:name="android.intent.action.MAIN" />`

menunjukkan bahwa aktivitas ini adalah titik awal yang dijalankan saat aplikasi dibuka pertama kali, lalu `<category android:name="android.intent.category.LAUNCHER" />` menjadikan aktivitas ini muncul di daftar aplikasi (launcher) pada perangkat Android. Saya juga menambahkan baris `<uses-permission android:name="android.permission.INTERNET"/>` yang merupakan deklarasi izin (permission) yang ditulis di dalam file `AndroidManifest.xml` untuk memberi tahu sistem Android bahwa aplikasi membutuhkan akses ke jaringan internet.

D. Tautan Git

https://github.com/dinzzti/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. Jawaban

RecyclerView masih banyak digunakan karena sudah terbukti stabil dan fleksibel, terutama dalam proyek besar atau aplikasi lama yang masih menggunakan sistem View XML. Meskipun LazyColumn dari Jetpack Compose punya kode yang lebih ringkas dan mudah dibaca, RecyclerView tetap unggul dalam hal kontrol detail, seperti pengelolaan animasi kompleks, dukungan untuk berbagai jenis item dalam satu daftar, serta integrasi dengan fitur-fitur canggih seperti drag-and-drop atau sticky headers. Selain itu, dalam pengujian performa, RecyclerView menunjukkan waktu startup yang lebih cepat dibandingkan LazyColumn, meskipun perbedaan ini dapat diminimalkan dengan optimisasi tambahan. Jadi, meskipun LazyColumn lebih sederhana dan cocok untuk proyek baru dengan Jetpack Compose, RecyclerView tetap menjadi pilihan yang solid untuk kebutuhan yang lebih kompleks atau proyek yang sudah ada.