LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 3

BUILD A SCROLLABLE LIST

Oleh:

Muhammad Daffa Musyafa NIM. 2310817110007

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 With Compose ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Prakitkum ini dikerjakan oleh:

Nama Praktikan : Muhammad Daffa Musyafa
NIM : 2310817210007

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

Soal Praktikum:

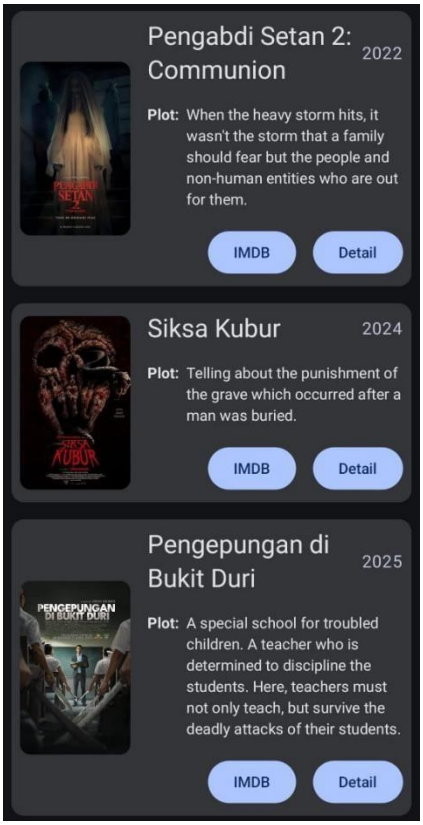
1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
   List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
   3.
   Item pada list 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
c. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
d. 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
e. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)

2. Aplikasi berbasis XML harus menggunakan ViewBinding

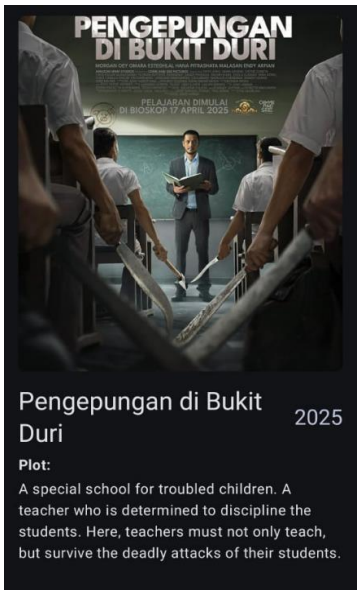
Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

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

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

A. Source Code XML

MainActivity.kt

Tabel 1 Source Code Jawaban soal 1 XML

```
1   package com.example.mobilelegendcharacterlistxml
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5
6
7   class MainActivity : AppCompatActivity() {
8
9       override fun onCreate(savedInstanceState: Bundle?) {
10          super.onCreate(savedInstanceState)
11          setContentView(R.layout.activity_main)
12
13          val fragmentManager = supportFragmentManager
14          val homeFragment = HomeFragment()
15          val fragment =
16  fragmentManager.findFragmentByTag(HomeFragment::class.java.simpleName
17  )
18          if (fragment !is HomeFragment) {
19              fragmentManager
20                  .beginTransaction()
21                  .add(R.id.frame_container, homeFragment,
22  HomeFragment::class.java.simpleName)
23                  .commit()
24          }
25      }
26  }
```

HeroML.kt

Tabel 2 Source Code Jawaban soal 1 XML

```
1   package com.example.mobilelegendcharacterlistxml
2   import android.os.Parcelable
3   import kotlinx.parcelize.Parcelize
4
5   @Parcelize
6   data class HeroML(
7       val name: String,
8       val image: Int,
9       val url: String,
10      val description: String
11  ): Parcelable
```

HeroMLAdapter.kt

Tabel 3 Source Code Jawaban soal 1 XML

```
1   package com.example.mobilelegendcharacterlistxml
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.RecyclerView
6   import
7   com.example.mobilelegendcharacterlistxml.databinding.ItemCharMlBindin
8   g
9
10  class HeroMLAdapter(
11      private val listHero: ArrayList<HeroML>,
12      private val onDetailClick: (String) -> Unit,
13      private val onPenjelasanClick: (String, Int, String) -> Unit)
14      : RecyclerView.Adapter<HeroMLAdapter.ListViewHolder>() {
15
16      inner class ListViewHolder (val binding: ItemCharMlBinding) :
17  RecyclerView.ViewHolder(binding.root){
18          fun bind(character : HeroML){
19              binding.tvItemName.text = character.name
20              binding.imgItemMl.setImageResource(character.image)
21              binding.tvIsi.text = character.description
22
23              binding.btnDescription.setOnClickListener {
24                  onDetailClick(character.url)
25              }
26
27              binding.btnPenjelasan.setOnClickListener {
28                  onPenjelasanClick(character.name,character.image
29  ,character.description)
30              }
31          }
32      }
33
34      override fun onCreateViewHolder(parent: ViewGroup, viewType:
35  Int): ListViewHolder {
36          val binding =
37  ItemCharMlBinding.inflate(LayoutInflater.from(parent.context),
38  parent, false)
39          return ListViewHolder(binding)
40      }
41
42      override fun getItemCount(): Int = listHero.size
43
44      override fun onBindViewHolder(holder: ListViewHolder, position:
45  Int) {
```

```
46          holder.bind(listHero[position])
47      }
48      }
```

HomeFragment.kt

Tabel 4 Source Code Jawaban soal 1 XML

```
1   package com.example.mobilelegendcharacterlistxml
2
3   import android.R.attr.description
4   import android.R.attr.name
5   import android.content.Intent
6   import android.net.Uri
7   import android.os.Bundle
8   import android.system.Os.link
9   import androidx.fragment.app.Fragment
10  import android.view.LayoutInflater
11  import android.view.View
12  import android.view.ViewGroup
13  import androidx.recyclerview.widget.LinearLayoutManager
14  import
15  com.example.mobilelegendcharacterlistxml.databinding.FragmentHomeBindi
16  ng
17
18  class HomeFragment : Fragment() {
19
20      private var _binding: FragmentHomeBinding? = null
21      private val binding get() = _binding!!
22
23      private lateinit var characterAdapter: HeroMLAdapter
24      private val list = ArrayList<HeroML>()
25
26
27      override fun onCreateView(
28          inflater: LayoutInflater, container: ViewGroup?,
29          savedInstanceState: Bundle?
30      ): View {
31          _binding = FragmentHomeBinding.inflate(inflater, container,
2   false)
33
34          list.clear()
35          list.addAll(getListHeroML())
36          setupRecyclerView()
37
38          return binding.root
39
40      }
41
42      private fun setupRecyclerView() {
```

```
43          characterAdapter = HeroMLAdapter(
44              list,
46              onDetailClick = { url ->
47                  val intent = Intent(Intent.ACTION_VIEW,
48  Uri.parse(url))
49                  startActivity(intent)
50              },
51              onPenjelasanClick = { name, image, description ->
52                  val detailFragment = DetailFragment().apply {
53                      arguments = Bundle().apply {
54                          putString("EXTRA_NAME", name)
55                          putInt("EXTRA_PHOTO", image)
56                          putString("EXTRA_DESCRIPTION", description)
57                      }
58                  }
59
60                  parentFragmentManager.beginTransaction()
61                      .replace(R.id.frame_container, detailFragment)
62                      .addToBackStack(null)
63                      .commit()
64              }
65          )
66
67          binding.rvCharacter.apply {
68              layoutManager = LinearLayoutManager(context)
69              adapter = characterAdapter
70              setHasFixedSize(true)
71          }
72      }
73
74      private fun getListHeroML(): ArrayList<HeroML> {
75          val dataName = resources.getStringArray(R.array.data_name)
76          val dataPhoto = resources.obtainTypedArray(R.array.data_photo)
77          val dataLink = resources.getStringArray(R.array.data_link)
78          val dataDesc = resources.getStringArray(R.array.data_desc)
79          val listCharacterML = ArrayList<HeroML>()
80          for (i in dataName.indices) {
81              val character =
82  HeroML(dataName[i],dataPhoto.getResourceId(i, -
83  1),dataLink[i],dataDesc[i])
84              listCharacterML.add(character)
85          }
86          dataPhoto.recycle()
87          return listCharacterML
88      }
89
90      override fun onDestroyView() {
91          super.onDestroyView()
92          _binding = null
93      }
94  }
```

DetailFragment.kt

Tabel 5 Source Code Jawaban soal 1 XML

```
1   package com.example.mobilelegendcharacterlistxml
2
3   import android.R.attr.text
4   import android.os.Bundle
5   import androidx.fragment.app.Fragment
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import
10  com.example.mobilelegendcharacterlistxml.databinding.FragmentDetailBin
11  ding
12
13
14  class DetailFragment : Fragment() {
15
16      private var _binding: FragmentDetailBinding? = null
17      private val binding get() = _binding!!
18
19
20      override fun onCreateView(
21          inflater: LayoutInflater, container: ViewGroup?,
22          savedInstanceState: Bundle?
23      ): View {
24          _binding = FragmentDetailBinding.inflate(inflater, container,
25  false)
26
27          val name = arguments?.getString("EXTRA_NAME")
28          val photo = arguments?.getInt("EXTRA_PHOTO")
29          val description = arguments?.getString("EXTRA_DESCRIPTION")
30
31
32          binding.tvItemName.text = name
33          photo?.let {
34              binding.imgItemMl.setImageResource(it)
35              binding.tvIsi.text = description
36          }
37
38          return binding.root
39      }
40
41      override fun onDestroyView() {
42          super.onDestroyView()
43          _binding = null
44      }
45  }
```

Dalam file layout:

activity_main.xml

Dalam XML ada beberapa file tambahan agar sama tampilannya dengan di gambar.

Tabel 6 Source Code Jawaban soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:id="@+id/frame_container"
8      tools:context=".MainActivity">
9  </FrameLayout>
```

item_char_ml.xml:

Tabel 7 Source Code Jawaban soal 1 XML

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.cardview.widget.CardView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:card_view="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:id="@+id/card_view"
7      android:layout_width="match_parent"
8      android:layout_height="wrap_content"
9      android:layout_gravity="center"
10     android:layout_marginStart="8dp"
11     android:layout_marginTop="4dp"
12     android:layout_marginEnd="8dp"
13     android:layout_marginBottom="4dp"
14     card_view:cardCornerRadius="4dp">
15
16
17     <androidx.constraintlayout.widget.ConstraintLayout
18         android:layout_width="match_parent"
19         android:layout_height="266dp"
20         android:padding="8dp">
21
22         <ImageView
23             android:id="@+id/img_item_ml"
24             android:layout_width="80dp"
25             android:layout_height="110dp"
26             android:scaleType="centerCrop"
27
```

```
card_view:layout_constraintBottom_toTopOf="@+id/btn_description"
            card_view:layout_constraintStart_toStartOf="parent"
            card_view:layout_constraintTop_toTopOf="parent" />


        <TextView
            android:id="@+id/tv_item_name"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:textSize="20sp"
            android:textStyle="bold"
            card_view:layout_constraintBottom_toTopOf="@+id/tv_isi"
            card_view:layout_constraintEnd_toEndOf="parent"

card_view:layout_constraintStart_toEndOf="@id/img_item_ml"
            card_view:layout_constraintTop_toTopOf="parent"
            card_view:layout_constraintVertical_bias="0.0"
            tools:text="Nama Hero" />

        <Button
            android:id="@+id/btn_description"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="164dp"
            android:layout_marginStart="20dp"
            android:text="Detail"
            card_view:layout_constraintStart_toStartOf="parent"

card_view:layout_constraintTop_toBottomOf="@id/tv_item_name" />

        <Button
            android:id="@+id/btn_penjelasan"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
            android:layout_marginTop="164dp"
            android:text="Role"

card_view:layout_constraintStart_toEndOf="@+id/btn_description"

card_view:layout_constraintTop_toBottomOf="@id/tv_item_name" />

        <TextView
            android:id="@+id/tv_isi"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="48dp"
            android:layout_marginStart="10dp"
            android:textSize="16sp"
            android:textStyle="bold"
            card_view:layout_constraintEnd_toEndOf="parent"
```

```
            card_view:layout_constraintHorizontal_bias="1.0"

card_view:layout_constraintStart_toEndOf="@+id/img_item_ml"
            card_view:layout_constraintTop_toTopOf="parent"

tools:text="jsndjnjkjnsjnsjidnjksniaunsijnjniskjnaksjnkjnkdjnkajsnkjd
hisnkjniaanskdnsihaksnidjhaksnidsjaksndisnaksjndisjnaksnjaklnjianskan
iusndsj" />

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

fragment_home.xml

Tabel 8 Source Code Jawaban soal 1 XML

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

    <!-- TODO: Update blank fragment layout -->
    <androidx.recyclerview.widget.RecyclerView
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:id="@+id/rv_character"
        android:layout_margin="15dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

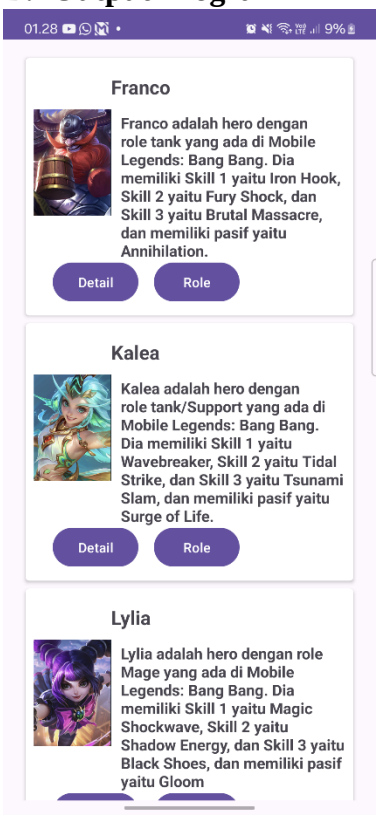
fragment_detail.xml

Tabel 9 Source Code Jawaban soal 1 XML

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

```

```
10      <ImageView
11          android:id="@+id/img_item_ml"
12          android:layout_width="100dp"
13          android:layout_height="150dp"
14          android:layout_marginTop="84dp"
15          android:scaleType="centerCrop"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          tools:src="@tools:sample/avatars" />
20
21      <TextView
21          android:id="@+id/tv_item_name"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          android:layout_marginTop="20dp"
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27          android:textSize="30dp"
28          app:layout_constraintTop_toBottomOf="@+id/img_item_ml"
29          tools:text="Nama Chara" />
30
31      <TextView
2           android:id="@+id/tv_isi"
33          android:layout_width="0dp"
34          android:layout_height="wrap_content"
35          android:layout_marginTop="32dp"
36          android:text="Deskripsi"
37          android:textSize="16sp"
38          app:layout_constraintEnd_toEndOf="parent"
39          app:layout_constraintHorizontal_bias="1.0"
40          app:layout_constraintStart_toStartOf="parent"
41          app:layout_constraintTop_toBottomOf="@+id/tv_item_name" />
42
43
44  </androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program



Gambar 3. Screenshot Hasil Jawaban Soal 1 XML

Gambar 4. Screenshot Hasil Jawaban Soal 1 XML

C. Pembahasan

MainActivity.kt:

Berikut adalah penjelasan kode yang kamu berikan dengan format yang sama seperti contohmu, sudah diberi jarak antarbagiannya, dan ditambah catatan jika ada hal yang perlu disiapkan di `build.gradle`:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3-4, dilakukan import terhadap `android.os.Bundle` yang digunakan untuk menyimpan dan meneruskan data antar lifecycle `Activity`, serta `androidx.appcompat.app.AppCompatActivity`, yaitu superclass dari activity yang mendukung fitur-fitur kompatibilitas ke versi Android lama menggunakan AndroidX.

Pada line 6–17, didefinisikan class `MainActivity` yang merupakan turunan dari `AppCompatActivity`, yaitu activity utama pada aplikasi ini. Di dalam method `onCreate`, pertama-tama dipanggil `super.onCreate(savedInstanceState)` untuk memanggil implementasi superclass dan menginisialisasi activity. Kemudian,

dipanggil `setContentView(R.layout.activity_main)` untuk menetapkan layout XML utama activity menggunakan file `activity_main.xml` sebagai tampilan UI.

Masih di dalam `onCreate`, objek `fragmentManager` diambil dari `supportFragmentManager`, yang digunakan untuk mengelola fragment dalam aplikasi. Sebuah instance dari `HomeFragment` dibuat dan disimpan dalam variabel `homeFragment`. Kemudian dilakukan pengecekan apakah fragment dengan tag `HomeFragment::class.java.simpleName` belum ditambahkan. Jika belum, maka fragment `HomeFragment` ditambahkan ke dalam layout dengan ID `frame_container` menggunakan `FragmentTransaction` dan ditandai dengan tag yang sama.

HeroML.kt
Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 2, diimpor `android.os.Parcelable`, yaitu interface yang digunakan untuk mengirim objek custom antar komponen Android (seperti antar Activity atau Fragment) melalui `Intent` atau `Bundle`.

Pada line 3, diimpor `kotlinx.parcelize.Parcelize`, yaitu annotation yang digunakan untuk menyederhanakan implementasi interface `Parcelable` tanpa perlu menulis kode boilerplate seperti `writeToParcel()` dan `describeContents()` secara manual.

Pada line 5–11, dideklarasikan data class `HeroML` yang menampung data karakter Mobile Legends. Kelas ini menggunakan anotasi `@Parcelize` untuk menggunakan `@Parcelize` harus mengubah gradle plugins `{id 'kotlin-parcelize'}` agar bisa otomatis diubah menjadi parcelable object. Properti-properti di dalam class ini terdiri dari:

- `name` bertipe `String` untuk menyimpan nama hero
- `image` bertipe `Int` untuk menyimpan ID resource gambar di drawable
- `url` bertipe `String` untuk menyimpan link yang berkaitan dengan hero tersebut
- `description` bertipe `String` untuk menyimpan deskripsi hero

Class ini mengimplementasikan `Parcelable` agar objeknya dapat dikirim lewat `Intent` atau disimpan dalam `Bundle` saat berpindah antar Fragment atau Activity.

HeroMLAdapter.kt
Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3–6, dilakukan import terhadap beberapa komponen penting yaitu `LayoutInflater` untuk mengubah file XML layout menjadi objek View, `ViewGroup` sebagai parent dari layout item di RecyclerView, serta `RecyclerView` dari library AndroidX yang berfungsi menampilkan daftar data secara efisien dan fleksibel. Kemudian di line 6, diimpor `ItemCharMlBinding`, yaitu kelas yang secara otomatis dihasilkan oleh ViewBinding berdasarkan file layout `item_char_ml.xml`. Kelas binding ini memudahkan akses ke view dalam layout XML tanpa perlu memanggil `findViewById()` secara manual, viewbinding ini perlu untuk menambah di gradle yaitu `viewBinding {enabled = true}`.

Pada line 8–11, didefinisikan class `HeroMLAdapter` yang merupakan turunan dari `RecyclerView.Adapter`. Adapter ini digunakan untuk mengatur tampilan daftar hero Mobile Legends. Konstruktor adapter menerima tiga parameter: `listHero`, yaitu daftar objek `HeroML` yang ingin ditampilkan; `onDetailClick`, yaitu lambda function yang dijalankan saat tombol "Description" diklik dan membawa data berupa `String` (URL hero); serta `onPenjelasanClick`, yaitu lambda function yang dijalankan saat tombol "Penjelasan" diklik dan membawa data berupa `String`, `Int`, dan `String` (nama, ID gambar, dan deskripsi hero).

Pada line 13–23, dideklarasikan inner class `ListViewHolder` yang mewarisi `RecyclerView.ViewHolder`. Kelas ini berisi fungsi `bind()` yang bertugas mengikat data dari objek `HeroML` ke view dalam layout. Komponen UI seperti `tvItemName`, `imgItemMl`, dan `tvIsi` masing-masing diatur untuk menampilkan nama hero, gambar hero menggunakan ID dari resource, serta deskripsi hero. Di dalam fungsi yang sama, listener ditambahkan pada dua tombol yaitu `btnDescription` dan `btnPenjelasan`. Tombol `btnDescription` akan memicu lambda `onDetailClick` dengan parameter URL hero, sedangkan tombol `btnPenjelasan` akan memicu lambda `onPenjelasanClick` dengan parameter nama, ID gambar, dan deskripsi hero tersebut.

Pada line 25–28, fungsi `onCreateViewHolder()` bertugas membuat instance dari `ListViewHolder`. Layout `item_char_ml.xml` di-*inflate* menggunakan `ItemCharMlBinding` untuk membuat tampilan setiap item pada RecyclerView. Binding ini kemudian digunakan untuk membentuk objek `ListViewHolder` yang akan merepresentasikan satu item hero.

Pada line 30, fungsi `getItemCount()` mengembalikan jumlah total data dalam `listHero`. Nilai ini menentukan berapa banyak item yang akan ditampilkan oleh RecyclerView.

Pada line 32–34, fungsi `onBindViewHolder()` dipanggil oleh RecyclerView untuk menampilkan data pada posisi tertentu. Fungsi ini akan mengambil data `HeroML` dari `listHero` berdasarkan indeks posisi, lalu memanggil fungsi `bind()` pada ViewHolder untuk menampilkannya ke dalam UI item tersebut.

HomeFragment.kt
Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlistxml`.

Pada line 3–11, dilakukan import terhadap berbagai komponen penting untuk membangun fragment, seperti `Intent`, `Uri`, `Bundle`, `Fragment`, `View`, `LayoutInflater`, `ViewGroup`, dan `LinearLayoutManager` dari `RecyclerView`. Selain itu, juga diimpor `FragmentHomeBinding` yang merupakan kelas ViewBinding otomatis dari layout XML `fragment_home.xml`. Namun, baris import `android.R.attr.description`, `android.R.attr.name`, dan `android.system.Os.link` sebetulnya **tidak diperlukan dan dapat dihapus**, karena tidak digunakan dalam kode dan bisa menimbulkan kebingungan karena berasal dari resource bawaan Android, bukan dari proyek aplikasi ini.

Pada line 13–16, didefinisikan class `HomeFragment` yang merupakan turunan dari `Fragment`. Fragment ini berfungsi menampilkan daftar karakter Mobile Legends dalam bentuk list menggunakan `RecyclerView`. Properti `_binding` digunakan sebagai tempat menyimpan binding terhadap layout, yang kemudian diakses aman melalui properti `binding`. Adapter untuk RecyclerView dideklarasikan dalam `characterAdapter`, dan daftar data hero dikelola melalui `list`, yang merupakan `ArrayList<HeroML>`.

Pada line 18–27, override dilakukan terhadap fungsi `onCreateView()` untuk meng-*inflate* layout fragment, mengisi daftar data hero dengan memanggil `getListHeroML()`, lalu menginisialisasi RecyclerView melalui fungsi `setupRecyclerView()`. Fungsi ini akan dijalankan saat tampilan fragment pertama kali dibuat, dan nilai kembaliannya adalah `binding.root`, yaitu root dari layout hasil ViewBinding.

Pada line 29–53, didefinisikan fungsi `setupRecyclerView()` yang digunakan untuk mengatur komponen RecyclerView di dalam fragment. Di dalamnya, `characterAdapter` diinisialisasi menggunakan `HeroMLAdapter`, dan dua lambda function diberikan untuk menangani tombol klik pada setiap item: tombol “Description” akan membuka link URL hero melalui `Intent` dengan `ACTION_VIEW`, sedangkan tombol “Penjelasan” akan mengganti fragment saat ini dengan `DetailFragment`, sambil meneruskan data `name`, `image`, dan `description` melalui `Bundle`. Fragment baru ditampilkan menggunakan `parentFragmentManager` dengan metode `replace()` ke dalam `R.id.frame_container`, dan transaksi disimpan ke back stack agar bisa dikembalikan.

Pada line 55–64, fungsi `getListHeroML()` didefinisikan untuk mengambil data dari resource berupa array. Data diambil dari `strings.xml` (untuk nama, link, dan deskripsi hero) dan `arrays.xml`/`typedArray` (untuk gambar). Seluruh data kemudian dikonversi menjadi list objek `HeroML`, lalu dikembalikan dalam bentuk `ArrayList`. Pemanggilan `dataPhoto.recycle()` di akhir berfungsi untuk membebaskan resource yang sudah tidak digunakan.

Pada line 66–68, fungsi `onDestroyView()` di-override untuk menghindari memory leak pada Fragment. Binding dihapus dengan mengatur `_binding = null` saat view fragment dihancurkan, sesuai praktik yang direkomendasikan saat menggunakan ViewBinding di dalam Fragment.

DetailFragment.kt
Pada line 1, dideklarasikan nama package file Kotlin yaitu com.example.mobilelegendcharacterlistxml.

Pada line 3–9, dilakukan import terhadap beberapa komponen penting seperti Bundle, Fragment, View, ViewGroup, LayoutInflater, dan FragmentDetailBinding. Namun, import android.R.attr.text pada baris 3 sebetulnya **tidak diperlukan dan bisa dihapus**, karena tidak digunakan dalam kode dan justru berpotensi menimbulkan konflik dengan resource aplikasi sendiri. FragmentDetailBinding adalah kelas otomatis yang dibuat Android Studio berdasarkan layout XML fragment_detail.xml dan hanya bisa digunakan jika ViewBinding telah diaktifkan dalam file Gradle.

Pada line 11–14, didefinisikan class DetailFragment yang merupakan turunan dari Fragment. Fragment ini bertugas untuk menampilkan detail dari karakter Mobile Legends yang dipilih. Seperti konvensi umum dengan ViewBinding di Fragment, digunakan properti _binding sebagai nullable, dan binding sebagai non-nullable untuk akses aman terhadap elemen UI setelah onCreateView() dipanggil.

Pada line 16–27, fungsi onCreateView() di-override untuk meng-*inflate* layout fragment_detail.xml menggunakan FragmentDetailBinding. Kemudian, data yang dikirim melalui arguments (berupa nama, gambar, dan deskripsi hero) diambil menggunakan getString() dan getInt(). Data ini kemudian digunakan untuk mengisi tampilan: nama karakter dimasukkan ke TextView tvItemName, gambar diatur ke ImageView imgItemMl jika tidak null, dan deskripsi dimasukkan ke TextView tvIsi. Hal ini memastikan tampilan detail sesuai dengan hero yang dipilih sebelumnya di HomeFragment.

Pada line 29–32, fungsi onDestroyView() di-override untuk menghindari memory leak. Binding dihapus dengan mengatur _binding = null saat fragment dihancurkan. Ini adalah praktik yang direkomendasikan oleh Google saat menggunakan ViewBinding di Fragment karena view Fragment memiliki siklus hidup yang berbeda dengan Fragment itu sendiri.

Activity_main.xml

Pada line 1–7, layout menggunakan `FrameLayout` dengan atribut `layout_width` dan `layout_height` diset ke `match_parent`, sehingga memenuhi seluruh layar perangkat. Elemen ini memiliki `id` yaitu `@+id/frame_container`, yang berfungsi sebagai referensi dalam `MainActivity.kt` untuk menambahkan fragment menggunakan metode `fragmentManager.beginTransaction().add(...).`

Atribut `tools:context=".MainActivity"` hanya digunakan saat design preview di Android Studio untuk memberi tahu bahwa layout ini digunakan oleh kelas `MainActivity`. Di dalam `FrameLayout` ini tidak ada elemen UI lain secara langsung karena kontennya akan diganti secara dinamis oleh fragment yang dimasukkan ke dalam `frame_container` tersebut saat runtime.

Item_char_ml.xml

File `item_char_ml.xml` merupakan layout yang digunakan sebagai tampilan item tunggal dalam `RecyclerView` di aplikasi ini. Layout ini digunakan di dalam `HeroMLAdapter.kt`, tepatnya di `ViewHolder` bernama `ListViewHolder`, untuk menampilkan setiap karakter Mobile Legends satu per satu dalam daftar.

Pada line 1–15, layout dibungkus oleh komponen `CardView` dari `androidx.cardview.widget.CardView`, yang memberikan efek bayangan dan sudut membulat pada item. Atribut `cardCornerRadius` diset ke 4dp agar sudutnya agak melengkung, dan terdapat margin di setiap sisi item agar tidak terlalu mepet satu sama lain ketika ditampilkan dalam daftar. `CardView` ini akan menjadi elemen visual utama dari setiap item hero.

Pada line 17–94, di dalam `CardView` terdapat `ConstraintLayout` yang digunakan untuk menyusun elemen-elemen UI secara fleksibel dengan constraint antar komponen. `ConstraintLayout` memiliki tinggi tetap 266dp dan padding 8dp agar isi tidak terlalu rapat ke tepi.

Di dalamnya, pertama ada `ImageView` dengan id `img_item_ml` (baris 19–26) yang digunakan untuk menampilkan gambar hero. Gambar ini berukuran 80dp x 110dp dan disetel dengan `scaleType="centerCrop"` agar gambar mengisi seluruh area dengan proporsional. Letaknya dikaitkan (`constraint`) di bagian atas layout dan sejajar secara vertikal dengan elemen lainnya.

Lalu ada `TextView` dengan id `tv_item_name` (baris 28–40) yang menampilkan nama hero. Lebarnya dibuat `0dp` karena menggunakan constraint start dan end, dengan ukuran teks 20sp dan gaya bold. Letaknya berada di atas elemen `tv_isi`, dan disesuaikan agar bersebelahan dengan `ImageView` sebelumnya.

Baris 42–54 Berikutnya terdapat dua buah `Button`, yaitu `btn_description` dan `btn_penjelasan`. Kedua tombol ini berada di bawah teks nama dan digunakan sebagai aksi untuk menampilkan link deskripsi hero (tombol Detail) dan penjelasan detail dalam fragment baru (tombol Role). Keduanya disejajarkan secara horizontal dengan sedikit jarak di antara keduanya.

Baris 56–68 Terakhir, ada `TextView` dengan id `tv_isi` yang berfungsi menampilkan deskripsi singkat dari hero tersebut. Komponen ini juga berada di samping `ImageView`, sejajar secara horizontal dan berada di atas tombol-tombol. Deskripsinya diset bold dengan ukuran 16sp agar mudah terbaca. Properti `tools:text` digunakan sebagai contoh isi deskripsi saat preview di Android Studio.

fragment_detail.xml

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8.

Pada line 2–5, digunakan `ConstraintLayout` sebagai root layout. Layout ini memungkinkan setiap elemen UI diposisikan relatif terhadap elemen lain dan/atau parent-nya. Layout memiliki lebar dan tinggi `match_parent`, sehingga memenuhi seluruh layar. Tiga namespace juga dideklarasikan dalam elemen root: `xmlns:android` digunakan untuk atribut standar Android seperti `layout_width`, `id`, dan sebagainya; `xmlns:tools` digunakan untuk kebutuhan preview di Android Studio, seperti memberikan data dummy melalui `tools:text`; dan `xmlns:app` digunakan untuk atribut-atribut khusus dari ConstraintLayout, seperti `app:layout_constraintTop_toBottomOf`.

Pada line 7–14, didefinisikan sebuah `ImageView` dengan ID `img_item_ml`. Komponen ini digunakan untuk menampilkan gambar karakter Mobile Legends. Ukurannya diset sebesar 100dp x 150dp dan `scaleType` diatur `centerCrop`, yang membuat gambar memenuhi seluruh area tampilan tanpa mengubah aspek rasio. Gambar ini diposisikan di tengah horizontal dengan margin atas sebesar 84dp dari parent layout.

Pada line 16–23, terdapat sebuah `TextView` dengan ID `tv_item_name` untuk menampilkan nama karakter. Ukuran teks dibuat cukup besar yaitu 30dp agar nama karakter lebih menonjol di layar. Elemen ini diletakkan tepat di bawah `ImageView` dengan margin atas 20dp, dan disejajarkan secara horizontal di tengah parent layout.

Pada line 25–32, dideklarasikan `TextView` kedua dengan ID `tv_isi`, yang digunakan untuk menampilkan deskripsi karakter Mobile Legends secara lebih lengkap. Lebarnya menggunakan `0dp` untuk mengikuti aturan `ConstraintLayout` (akan dihitung berdasarkan batas kiri dan kanan), dengan tinggi menyesuaikan isi. Posisi elemen ini berada di bawah nama karakter dengan margin atas 32dp. Ukuran teks disesuaikan agar tetap nyaman dibaca, yaitu 16sp, dan penempatan horizontalnya tetap berada di tengah lebar layout.

fragment_char.xml

Pada line 1, dideklarasikan bahwa file ini merupakan file XML dengan encoding UTF-8. Ini adalah deklarasi standar untuk file XML agar sistem dapat memahami format karakter yang digunakan.

Pada line 2–7, digunakan `ConstraintLayout` sebagai root layout. `ConstraintLayout` merupakan jenis layout fleksibel yang memungkinkan setiap elemen UI diposisikan secara relatif terhadap elemen lain maupun terhadap parent-nya. Layout ini memiliki lebar dan tinggi `match_parent`, yang berarti akan memenuhi seluruh ukuran layar. Tiga namespace juga didefinisikan di sini:

Tiga namespace didefinisikan dalam elemen root `ConstraintLayout`. `xmlns:android` digunakan untuk atribut umum Android seperti `layout_width`, `id`, dan lainnya. `xmlns:tools` digunakan oleh Android Studio untuk keperluan preview layout, seperti menampilkan data dummy saat proses desain. Sementara itu, `xmlns:app` dipakai untuk atribut khusus yang berasal dari library AndroidX, termasuk atribut-atribut dari `ConstraintLayout` seperti `app:layout_constraintTop_toTopOf`.

Pada line 9–17, dideklarasikan sebuah komponen `RecyclerView` dengan ID `rv_character`. Komponen ini digunakan untuk menampilkan daftar karakter Mobile Legends dalam bentuk list atau grid yang bisa discroll. Lebar dan tinggi diatur `0dp`, yang berarti mengikuti aturan constraint dari `ConstraintLayout`. RecyclerView ini diberi margin sebesar 15dp di keempat sisi agar tidak menempel langsung ke tepi layar. Untuk posisi atributnya `app:layout_constraintTop_toTopOf="parent"` dan `app:layout_constraintBottom_toBottomOf="parent"` digunakan untuk menempatkan komponen dari bagian atas hingga bawah, sehingga memenuhi tinggi parent-nya secara vertikal. Sedangkan `app:layout_constraintStart_toStartOf="parent"` dan `app:layout_constraintEnd_toEndOf="parent"` membuat `RecyclerView` memanjang secara horizontal dari kiri ke kanan, mengikuti lebar parent. Dengan keempat constraint ini, `RecyclerView` akan ditampilkan memenuhi seluruh layar, dengan margin di sekelilingnya yang telah ditentukan melalui atribut `android:layout_margin`.

D. Source Code Compose
MainActivity.kt

Tabel 10 Source Code Jawaban soal 1 Compose

```
1   package com.example.mobilelegendcharacterlist
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.enableEdgeToEdge
7   import androidx.compose.foundation.Image
8   import androidx.compose.foundation.layout.*
9   import androidx.compose.foundation.lazy.LazyColumn
10  import androidx.compose.foundation.shape.RoundedCornerShape
11  import androidx.compose.material3.*
12  import androidx.compose.runtime.Composable
13  import androidx.compose.ui.Alignment
14  import androidx.compose.ui.Modifier
15  import androidx.compose.ui.platform.LocalContext
16  import androidx.compose.ui.res.painterResource
17  import androidx.compose.ui.text.font.FontWeight
18  import androidx.compose.ui.text.style.TextOverflow
19  import androidx.compose.ui.tooling.preview.Preview
20  import androidx.compose.ui.unit.dp
21  import androidx.compose.ui.unit.sp
22  import
23  com.example.mobilelegendcharacterlist.ui.theme.MobileLegendCharacterLi
24  stTheme
25  import com.example.mobilelegendcharacterlist.heroListML.heroList
26  import android.content.Intent
27  import android.net.Uri
28  import androidx.navigation.NavHostController
29  import androidx.navigation.NavType
30  import androidx.navigation.compose.NavHost
31  import androidx.navigation.compose.composable
32  import androidx.navigation.compose.rememberNavController
33  import androidx.navigation.navArgument
34
35  class MainActivity : ComponentActivity() {
36      override fun onCreate(savedInstanceState: Bundle?) {
37          super.onCreate(savedInstanceState)
38          enableEdgeToEdge()
39          setContent {
40              MobileLegendCharacterListTheme {
41                  Surface(
42                      modifier = Modifier.fillMaxSize(),
43                      color = MaterialTheme.colorScheme.background
44                  ) {
45                      val navController = rememberNavController()
46                      NavHost(
```

```
                        navController = navController,
                        startDestination = "heroList"
                    ) {
                        composable("heroList") {
                            HeroList(navController)
                        }
                        composable("penjelasan/{description}/{image}",
                            arguments = listOf(
                                navArgument("description") { type =
NavType.StringType },
                                navArgument("image") { type =
NavType.IntType }
                            )
                        ) { backStackEntry ->
                            val description =
backStackEntry.arguments?.getString("description") ?: ""
                            val image =
backStackEntry.arguments?.getInt("image") ?: 0
                            PenjelasanScreen(description, image)
                        }
                        }
                    }
                }
            }
        }
    }

    @Composable
    fun HeroList(navController: NavHostController) {
        LazyColumn(
            modifier = Modifier
                .fillMaxWidth()
                .padding(20.dp)
        ) {
            items(heroList.size) { DataHero ->
                val heroes = heroList[DataHero]
                HeroItem(
                    name = heroes.name,
                    image = heroes.image,
                    url = heroes.url,
                    description = heroes.description,
                    navController = navController
                )
            }
        }
    }

    @Composable
    fun HeroItem(name: String, image: Int, url: String, description:
String, navController: NavHostController) {
        val context = LocalContext.current
        Card(
```

```
            modifier = Modifier
                .fillMaxWidth()
                .padding(10.dp),
            shape = RoundedCornerShape(16.dp),
            elevation = CardDefaults.cardElevation(defaultElevation =
4.dp)
        ) {
            Row(
                modifier = Modifier
                    .padding(16.dp)
                    .fillMaxWidth(),
                verticalAlignment = Alignment.CenterVertically
            ) {
                Image(
                    painter = painterResource(id = image),
                    contentDescription = null,
                    modifier = Modifier
                        .size(width = 100.dp, height = 120.dp)
                )
                Spacer(modifier = Modifier.width(16.dp))
                Column(
                    modifier = Modifier
                        .weight(1f)
                ) {
                    Text(text = name, fontWeight = FontWeight.Bold,
fontSize = 20.sp)
                    Spacer(modifier = Modifier.height(4.dp))
                    Text(
                        text = description,
                        fontSize = 14.sp,
                        maxLines = 3,
                        overflow = TextOverflow.Ellipsis
                    )
                    Spacer(modifier = Modifier.height(8.dp))
                    Row(
                        horizontalArrangement =
Arrangement.SpaceBetween,
                        modifier = Modifier
                            .fillMaxWidth()
                            .wrapContentWidth(Alignment.Start),
                    ) {
                        Button(
                            onClick = {
                                val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(url))
                                context.startActivity(intent)
                            },
                            contentPadding = PaddingValues(horizontal
= 16.dp, vertical = 8.dp),
                            shape = RoundedCornerShape(50),
                            modifier =
Modifier.defaultMinSize(minWidth = 1.dp)
```

```
151                         ) {
152                             Text("Detail", fontSize = 14.sp)
153                         }
154                         Spacer(modifier = Modifier.width(8.dp))
155                         Button(
156                             onClick = {
157                                 val encodedDesc =
158 Uri.encode(description)
159 
160 navController.navigate("penjelasan/${Uri.encode(description)}/$image")
161                             },
162                             contentPadding = PaddingValues(horizontal
163 = 16.dp, vertical = 8.dp),
164                             shape = RoundedCornerShape(50),
165                             modifier =
166 Modifier.defaultMinSize(minWidth = 1.dp)
167                         ) {
168                             Text("Penjelasan", fontSize = 13.sp)
169                         }
170                     }
171                 }
172             }
173         }
174     }
175 
176     @Composable
177     fun PenjelasanScreen(description: String, image: Int) {
178         Column(
179             modifier = Modifier
180                 .fillMaxSize()
181                 .padding(16.dp)
182                 .padding(WindowInsets.statusBars.asPaddingValues()),
183             horizontalAlignment = Alignment.CenterHorizontally
184         ) {
185             Image(
186                 painter = painterResource(id = image),
187                 contentDescription = null,
188                 modifier = Modifier
189                     .fillMaxWidth()
190                     .height(200.dp)
191             )
192             Spacer(modifier = Modifier.height(16.dp))
193             Text(
194                 text = description,
195                 fontSize = 16.sp
196             )
197         }
198     }
199 
200     @Preview(showBackground = true)
201     @Composable
202     fun GreetingPreview() {
```

```
        MobileLegendCharacterListTheme {
            Text("Preview List Hero")
        }
}
```

dataList.kt

Tabel 11 Source Code Jawaban soal 1 Compose

```
package com.example.mobilelegendcharacterlist.mobileLegendDataList

data class DataHero(
    val name: String,
    val image: Int,
    val url: String,
    val description : String
)
```

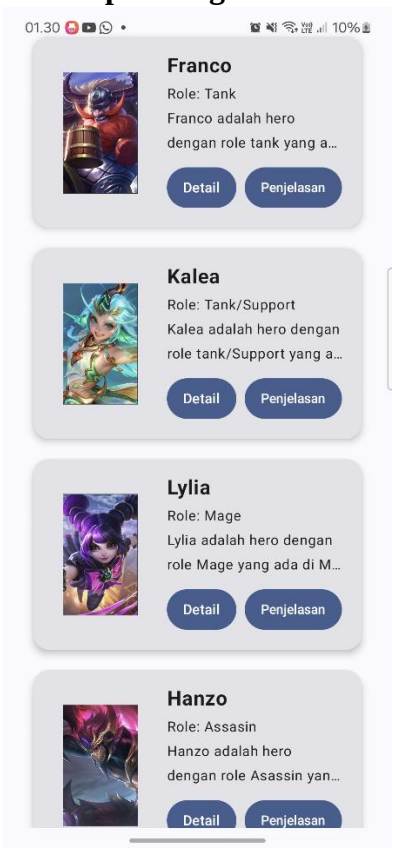
dataList.kt

Tabel 12 Source Code Jawaban soal 1 Compose

```
package com.example.mobilelegendcharacterlist.heroListML


import com.example.mobilelegendcharacterlist.R
import
com.example.mobilelegendcharacterlist.mobileLegendDataList.DataHero

val heroList = listOf(
    DataHero(name ="Franco",
        R.drawable.hero1,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2678746&heroid=10
",
        description = "Role: Tank\n" +
                "Franco adalah hero dengan role tank yang ada di
Mobile Legends: Bang Bang. Dia memiliki Skill 1 yaitu 'Iron Hook',
Skill 2 yaitu 'Fury Shock', dan Skill 3 yaitu 'Brutal Massacre', dan
memiliki pasif yaitu 'Annihilation'."),
    DataHero(name ="Kalea",
        R.drawable.hero2,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2863075&heroid=12
8",
        description = "Role: Tank/Support\n" +
                "Kalea adalah hero dengan role tank/Support yang ada
di Mobile Legends: Bang Bang. Dia memiliki Skill 1 yaitu
'Wavebreaker', Skill 2 yaitu 'Tidal Strike', dan Skill 3 yaitu
```

```
'Tsunami Slam', dan memiliki pasif yaitu 'Surge of Life'."),
    DataHero(name ="Lylia",
        R.drawable.hero3,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2678822&heroid=86
",
        description = "Role: Mage\n" +
                "Lylia adalah hero dengan role Mage yang ada di Mobile
Legends: Bang Bang. Dia memiliki Skill 1 yaitu 'Magic Shockwave',
Skill 2 yaitu 'Shadow Energy', dan Skill 3 yaitu 'Black Shoes', dan
memiliki pasif yaitu 'Angry Gloom'."),
    DataHero(name ="Hanzo",
        R.drawable.hero4,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2678805&heroid=69
",
        description = "Role: Assasin\n" +
                "Hanzo adalah hero dengan role Asassin yang ada di
Mobile Legends: Bang Bang. Dia memiliki Skill 1 yaitu 'Ninjutsu: Demon
Feast', Skill 2 yaitu 'Ninjutsu: Dark Mist', dan Skill 3 yaitu
'Kinjutsu: Pinnacle Ninja', dan memiliki pasif yaitu 'Ame no
Habakiri'."),
    DataHero(name ="Lancelot",
        R.drawable.hero5,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2678783&heroid=47
",
        description = "Role: Assasin\n" +
                "Lancelot adalah hero dengan role Assasin yang ada di
Mobile Legends: Bang Bang. Dia memiliki Skill 1 yaitu 'Puncture',
Skill 2 yaitu 'Thorned Rose', dan Skill 3 yaitu 'Phantom Execution',
dan memiliki pasif yaitu 'Soul Cutter'."),
    DataHero(name ="Lukas",
        R.drawable.hero6,
        url =
"https://www.mobilelegends.com/hero/detail?channelid=2819992&heroid=12
7",
        description = "Role: Fighter\n" +
                "Lukas adalah hero dengan role tank yang ada di Mobile
Legends: Bang Bang. Dia memiliki Skill 1 yaitu 'Flash Combo', Skill 2
yaitu 'Flash Step', dan Skill 3 yaitu 'Unleash the Beast', dan
memiliki pasif yaitu 'Hero's Resolve'.")

)
```

E. Output Program



Gambar 5 Screenshot Hasil Jawaban Soal 1

Gambar 6 Screenshot Hasil Jawaban Soal 1

F. Pembahasan

MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlist`.

Pada line 3–22, dilakukan import terhadap berbagai komponen dan library yang dibutuhkan dalam Jetpack Compose, seperti `Image`, `Column`, `Text`, `LazyColumn`, `Card`, `Button`, `TextOverflow`, dan `Modifier`. Selain itu juga mengimpor resource seperti `painterResource`, tools preview, serta dependensi untuk navigasi seperti `NavHostController`, `NavType`, dan `composable`. Ini memungkinkan kita untuk membangun UI deklaratif dan navigasi antar layar.

Pada line 24–42, didefinisikan class `MainActivity` yang merupakan turunan dari `ComponentActivity`, yaitu activity dasar untuk aplikasi yang menggunakan Jetpack Compose. Di dalam fungsi `onCreate()`, dipanggil `enableEdgeToEdge()` agar aplikasi tampil full screen pada device modern. Kemudian `setContent` digunakan untuk mengatur isi tampilan dari aplikasi, yang dibungkus dalam tema `MobileLegendCharacterListTheme`. Di dalamnya dibuat instance

`NavHostController` dan didefinisikan `NavHost` dengan dua composable: `heroList` untuk tampilan daftar hero, dan `penjelasan/{description}/{image}` untuk menampilkan penjelasan hero berdasarkan parameter yang dikirim melalui navigasi.

Pada line 44–54, didefinisikan fungsi composable `HeroList()` yang menampilkan daftar hero menggunakan `LazyColumn`. Setiap item dalam list diambil dari `heroList` dan akan dirender menggunakan fungsi `HeroItem()`. Fungsi ini menerima `navController` sebagai parameter agar bisa melakukan navigasi ke halaman penjelasan saat tombol ditekan.

Pada line 56–98, didefinisikan fungsi composable `HeroItem()` yang menampilkan setiap karakter/hero dalam bentuk card. Di dalam card ditampilkan gambar, nama hero, dan deskripsi singkat yang dibatasi 3 baris (dengan `maxLines = 3` dan `overflow = TextOverflow.Ellipsis`). Terdapat dua tombol: tombol Detail yang membuka URL hero di browser menggunakan `Intent`, dan tombol Penjelasan yang menavigasi ke layar penjelasan dengan parameter `description` dan `image`. `Uri.encode()` digunakan untuk menghindari error saat ada karakter khusus di URL atau deskripsi.

Pada line 100–110, didefinisikan fungsi composable `PenjelasanScreen()` yang menerima dua parameter: `description` dan `image`. Di dalamnya ditampilkan gambar hero dan deskripsi lengkap yang ditampilkan dalam layout `Column` dengan padding serta responsif terhadap status bar (menggunakan `WindowInsets.statusBars.asPaddingValues()`).

Pada line 112–115, didefinisikan fungsi preview `GreetingPreview()` yang akan ditampilkan di Android Studio Preview. Ini hanya menampilkan `Text("Preview List Hero")` sebagai placeholder preview agar bisa melihat hasil tampilan saat sedang mengembangkan aplikasi tanpa harus menjalankan emulator.

dataList.kt

Pada line 1, dideklarasikan nama package yaitu `com.example.mobilelegendcharacterlist.mobileLegendDataList`, yang merupakan sub-package dari aplikasi. Package ini berfungsi untuk mengelompokkan file-file yang berkaitan dengan data list hero Mobile Legends agar lebih terorganisasi.

Pada line 3, didefinisikan sebuah `data class` bernama `DataHero`. `data class` di Kotlin secara otomatis menyediakan fungsi-fungsi penting seperti `toString()`, `equals()`, `hashCode()`, dan `copy()`, yang berguna untuk menyimpan dan mengelola data secara efisien.

Data class `DataHero` merepresentasikan satu entitas hero Mobile Legends dan memiliki empat properti:

Pada line 4, properti `name` bertipe `String`, menyimpan nama hero seperti "Franco", "Kalea", dll.

Pada line 5, properti `image` bertipe `Int`, menyimpan ID dari resource drawable (biasanya `R.drawable.nama_gambar`) yang digunakan untuk menampilkan gambar hero.

Pada line 6, properti `url` bertipe `String`, menyimpan tautan eksternal (link) resmi atau sumber informasi tentang hero tersebut, yang akan dibuka menggunakan browser ketika tombol "Detail" ditekan.

Pada line 7, properti `description` bertipe `String`, berisi penjelasan atau informasi lengkap mengenai hero, yang akan ditampilkan pada halaman `PenjelasanScreen`.

HeroMLAdapter.kt

Pada line 1, dideklarasikan nama package file Kotlin yaitu `com.example.mobilelegendcharacterlist.heroListML`, yang menunjukkan bahwa file ini berada di dalam folder `heroListML` dari package utama aplikasi.

Pada line 3–5, dilakukan import terhadap resource dari file XML melalui `com.example.mobilelegendcharacterlist.R`, serta import class `DataHero` dari package `mobileLegendDataList`. Class `DataHero` kemungkinan besar didefinisikan sebagai data class yang merepresentasikan entitas hero Mobile Legends dengan properti seperti `name`, `imageResId`, `url`, dan `description`. Class ini berfungsi sebagai struktur data utama yang akan digunakan untuk menampilkan daftar hero pada tampilan RecyclerView atau Fragment detail.

Pada line 7–25, didefinisikan sebuah list bernama `heroList` yang berisi kumpulan objek `DataHero`. Setiap objek mewakili satu hero Mobile Legends dan memuat informasi lengkap seperti nama hero, ID gambar dari resource drawable (`R.drawable.hero1`, `hero2`, dan seterusnya), URL resmi untuk detail hero dari situs Mobile Legends, serta deskripsi singkat yang memuat role dan kemampuan dari hero tersebut. Deskripsi ditulis dengan format string multiline menggunakan karakter `\n` untuk membuat baris baru agar mudah dibaca di tampilan aplikasi. Seluruh data disimpan secara hardcoded dalam list menggunakan fungsi `listOf()`, sehingga data ini bersifat statis dan tidak dinamis dari API atau database.

Untuk dapat menggunakan gambar-gambar hero ini, kamu harus memastikan bahwa semua file gambar (`hero1.png`, `hero2.png`, dan seterusnya) sudah dimasukkan ke dalam folder `res/drawable` di project Android Studio kamu. Jika tidak, maka aplikasi akan mengalami error saat mencoba memuat resource tersebut.

Soal 2

RecyclerView masih digunakan karena banyak aplikasi lama yang dibangun dengan View XML dan belum bermigrasi ke Jetpack Compose, RecyclerView menawarkan kontrol lebih mendetail terhadap tampilan list, seperti pengaturan layout, animasi, dan dekorasi item. Banyak library pihak ketiga juga masih bergantung pada RecyclerView. Meskipun LazyColumn di Jetpack Compose lebih ringkas dan modern, migrasi penuh memerlukan waktu dan sumber daya, sehingga RecyclerView tetap relevan di banyak proyek.

Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/Easydaf/Praktikum_Mobile/tree/main/Modul3