

**LAPORAN PRAKTIKUM  
PEMROGRAMAN MOBILE  
MODUL 3**



**BUILD A SCROLLABLE LIST**

**Oleh:**

**Muhammad Azwin Hakim**

**NIM. 2310817310012**

**PROGRAM STUDI TEKNOLOGI INFORMASI  
FAKULTAS TEKNIK  
UNIVERSITAS LAMBUNG MANGKURAT  
APRIL 2024**

**LEMBAR PENGESAHAN**  
**LAPORAN PRAKTIKUM PEMROGRAMAN I**  
**MODUL 3**

Laporan Praktikum Pemrograman Mobile Modul 3: Build a Scrollable List ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Azwin Hakim  
NIM : 2310817310012

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

LEMBAR PENGESAHAN .....	2
DAFTAR ISI .....	3
DAFTAR GAMBAR.....	4
DAFTAR TABEL .....	5
SOAL 1 .....	6
A.    Source Code .....	8
B.    Output Program.....	17
C.    Pembahasan .....	19
D.    Tautan Git .....	22

## DAFTAR GAMBAR

Gambar 1. Contoh UI List .....	7
Gambar 2. Contoh UI Detail .....	7
Gambar 3. Screenshot Hasil Jawaban Soal 1 .....	17
Gambar 4. Screenshot Hasil Jawaban Soal 1 .....	18
Gambar 5. Screenshot Hasil Jawaban Soal 1 .....	18

## DAFTAR TABEL

Tabel 1. Source Code Jawaban Soal 1.....	8
Tabel 2. Source Code Jawaban Soal 1.....	9
Tabel 3. Source Code Jawaban Soal 1.....	11
Tabel 4. Source Code Jawaban Soal 1.....	14
Tabel 5. Source Code Jawaban Soal 1.....	16

## SOAL 1

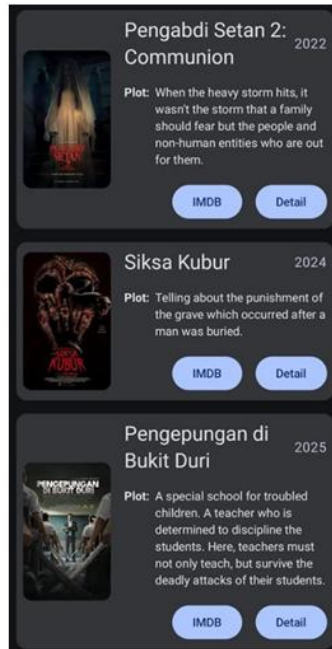
### Soal Praktikum:

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:

1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
  - a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
  - b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
7. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)
8. Aplikasi berbasis XML harus menggunakan ViewBinding

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

UI item list harus berisi 1 gambar, 2 button (intent eksplisit dan navigasi), dan 2 baris teks dan setiap baris memiliki 2 teks yang berbeda. Diusahakan agar desain UI item list menyerupai UI berikut:



Gambar 1. Contoh UI List

Desain UI laman detail bebas, tetapi diusahakan untuk mengikuti kaidah desain Material Design dan data item ditampilkan penuh di laman detail seperti contoh berikut:



Gambar 2. Contoh UI Detail

## A. Source Code

### 1. MainActivity.kt

```
1 package com.example.ultraman
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.core.view.WindowCompat
7 import androidx.compose.runtime.Composable
8 import androidx.navigation.NavHostController
9 import androidx.navigation.compose.NavHost
10 import androidx.navigation.compose.composable
11 import androidx.navigation.compose.rememberNavController
12 import androidx.navigation.NavType
13 import androidx.navigation.navArgument
14 import com.example.ultraman.ui.screens.DetailScreen
15 import com.example.ultraman.ui.screens.ListScreen
16 import com.example.ultraman.ui.theme.UltramanTheme
17
18 class MainActivity : ComponentActivity() {
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21         WindowCompat.setDecorFitsSystemWindows(window, false)
22         setContent {
23             UltramanTheme {
24                 val navController = rememberNavController()
25                 AppNavHost(navController)
26             }
27         }
28     }
29 }
30
31 @Composable
32 fun AppNavHost(navController: NavHostController) {
33     NavHost(navController = navController, startDestination =
34     "list") {
35         composable("list") {
36             ListScreen(navController)
37         }
38         composable(
39             "detail/{itemId}",
40             arguments = listOf(navArgument("itemId") { type =
41             NavType.IntType })
42             ) { backStackEntry ->
43                 val itemId =
44                 backStackEntry.arguments?.getInt("itemId")
45                 DetailScreen(itemId)
46             }
47     }
48 }
```

Tabel 1. Source Code Jawaban Soal 1



## 2. ListUltraman.kt

```
1 package com.example.ultraman.model
2
3 data class ListUltraman(
4     val id: Int,
5     val Name: String,
6     val HumanHost: String,
7     val Height: String,
8     val Weight: String,
9     val Detail: String,
10    val imageUrl: String,
11    val link: String
12 )
```

Tabel 2. Source Code Jawaban Soal 1

## 3. DataUltraman.kt

```
1 package com.example.ultraman
2
3 import com.example.ultraman.model.ListUltraman
4
5 val ultramans = listOf(
6     ListUltraman(
7         id = 1,
8         name = "Ultraman Noa",
9         host = "Kazuki Komon",
10        height = "55 m",
11        weight = "55,000 t",
12        description = "Ultraman Noa (ウルトラマンノア, Urutoraman Noa) is an ancient Ultraman from the World of N who pursued his evil counterpart, Dark Zagi, throughout various universes and countless alien worlds, including the World of the Land of Light. He is an Ultra who has been active across the multiverse gaining the moniker of a legendary figure for his power and feats. Noa is also the true form of Ultraman the Next/Ultraman Nexus.",
13        imageUrl = "https://th.bing.com/th/id/OIP.1rd_goNDK6agElvRe4AMyAHaKX?rs=1&pid=ImgDetMain",
14        wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Noa"
15    ),
16    ListUltraman(
17        id = 2,
18        name = "Ultraman Nexus",
19        host = "Kazuki Komon",
20        height = "49 m",
21        weight = "40,000 t",
22        description = "Ultraman Nexus (ウルトラマンネクサス, Urutoraman Nekusasu) is Ultraman Noa's second devolved form,
```

8	<p>and is the form a Dunamist assumes after using the Evoltruster. Starring in Ultraman Nexus, his story represented as the final phase of Ultra N Project. After evolving from Ultraman the Next, Nexus would resurface in 2008 to fight against a wave of Space Beast assaults, slowly growing in strength after bonding with multiple Dunamists and finally regaining his long lost form Ultraman Noa in his battle with Dark Zagi.",</p> <pre>         imageUrl = "https://pbs.twimg.com/media/FVhSSBHagAAWSju.jpg",         wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Nexus_(character)"     ),     ListUltraman(         id = 3,         name = "Ultraman Mebius",         host = "Mirai Hibino",         height = "49 m",         weight = "35,000 t",         description = "Ultraman Mebius (ウルトラマンメビウス, Urutoraman Mebiusu) is the protagonist of the eponymous series Ultraman Mebius. He was sent to Earth 25 years after Ultraman 80 to protect the planet from a new wave of monsters and aliens, many of which arrived due to Alien Empera's resurgence. During his tenure on Earth, he took the form of Mirai Hibino, and joined Crew GUYS.",         imageUrl = "https://pbs.twimg.com/media/FVhVh22akAA69s3.jpg",         wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Mebius_(character)"     ),     ListUltraman(         id = 4,         name = "Ultraman Tiga",         host = "Daigo Madoka",         height = "Micro ~ 53 m",         weight = "44,000 t",         description = "Ultraman Tiga (ウルトラマンティガ, Urutoraman Tiga) is the eponymous Ultra Hero of Ultraman Tiga. He is notable for being the first Ultraman in the franchise to be able to change forms, as well as being the first Ultra to appear in a televised series since Ultraman 80, ending a 15- year semi-hiatus.",         imageUrl = "https://images-wixmp- ed30a86b8c4ca887773594c2.wixmp.com/f/aad45e4a-312a-469e-b82d- 3dc010e4a687/dfofafd-cc54df22-844a-424d-b730- af3c8caee3d3.jpg/v1/fill/w_1280,h_2157,q_75,strp/ultraman_tig a_dark_and_multi_by_ultrahorizongax2021_dfofafd- fullview.jpg",         wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Tiga_(character)"     ), </pre>
9	

10	<pre> ListUltraman(     id = 5,     name = "Ultraman Belial",     host = "Arie Ishikari",     height = "55 m",     weight = "60,000 t",     description = "Ultraman Belial (ウルトラマンベリアル, Urutoraman Beriaru) was an evil Ultraman from the Land of Light, who was best known as Ultraman Zero's arch enemy and the father of Ultraman Geed. He was once a great fighter who took part in the Ultimate Wars, but his pride and greed got the better of him. This ultimately led him to commit the heinous crime of attempting to steal the Plasma Spark, for which he was banished. Injured and cast out, Belial was approached and transformed by Alien Reiblood into a Reionics, forever turning him to the darkness.\n" + </pre>
11	<pre> "Belial seemingly met his final end at the hands of his synthetic son, Ultraman Geed, in a climactic final battle that decided the fate of the universe. However, his existence continues to send ripples through the multiverse in the form of the Devil Splinters derived from his cells, as well as a parallel version of himself who lives on in the present day, having had his fate forever changed by a chance encounter with Absolute Tartarus.", "https://tsuburaya-prod.com/wp- content/uploads/2019/12/berial2.jpg", "https://ultra.fandom.com/wiki/Ultraman_Belial"), </pre>
12	<pre> ListUltraman(     id = 6,     name = "Ultraman Zero",     host = "Run (ラン, Ran)",     height = "Micro ~ 49 m",     weight = "35,000 t",     description = "Ultraman Zero (ウルトラマンゼロ, Urutoraman Zero) is the son of Ultraseven. He was trained under Ultraman Leo after he was banished from the Land of Light by his father for attempting to take the Plasma Spark for himself. His training ultimately led to his redemption in his battle with Belial.\n" + </pre>
13	<pre> "Zero is one of the most popular Ultras in the Ultraman Series' history, appearing many times over a decade since his debut.", "https://th.bing.com/th/id/OIP.E9iggNICwtyI7UR4WSD- 1wHaKm?rs=1&amp;pid=ImgDetMain", </pre>
14	<pre> "https://ultra.fandom.com/wiki/Ultraman_Zero") ) </pre>

Tabel 3. Source Code Jawaban Soal 1

#### 4. ListScreen.kt

1	package com.example.ultraman.ui.screens
2	

```

3 import android.content.Intent
4 import androidx.compose.foundation.layout.Arrangement
5 import androidx.compose.foundation.layout.Column
6 import androidx.compose.foundation.layout.Row
7 import androidx.compose.foundation.layout.Spacer
8 import androidx.compose.foundation.layout.WindowInsets
9 import androidx.compose.foundation.layout.fillMaxSize
10 import androidx.compose.foundation.layout.fillMaxWidth
11 import androidx.compose.foundation.layout.height
12 import androidx.compose.foundation.layout.padding
13 import androidx.compose.foundation.layout.systemBars
14 import androidx.compose.foundation.layout.asPaddingValues
15 import androidx.compose.ui.text.style.TextOverflow
16 import androidx.compose.foundation.layout.width
17 import androidx.compose.foundation.lazy.LazyColumn
18 import androidx.compose.foundation.shape.RoundedCornerShape
19 import androidx.compose.material3.Button
20 import androidx.compose.material3.Card
21 import androidx.compose.material3.CardDefaults
22 import androidx.compose.material3.MaterialTheme
23 import androidx.compose.material3.Text
24 import androidx.compose.runtime.Composable
25 import androidx.compose.ui.Modifier
26 import androidx.compose.ui.draw.clip
27 import androidx.compose.ui.layout.ContentScale
28 import androidx.compose.ui.unit.dp
29 import androidx.navigation.NavHostController
30 import androidx.compose.foundation.lazy.items
31 import androidx.compose.ui.text.font.FontWeight
32 import com.example.ultraman.ultramans
33 import androidx.core.net.toUri
34
35 @Composable
36 fun ListScreen(navController: NavHostController) {
37     LazyColumn(
38         modifier = Modifier
39             .fillMaxSize()
40             .padding(8.dp)
41
42         .padding(WindowInsets.systemBars.asPaddingValues())
43     ) {
44         items(ultramans) { item ->
45             Card(
46                 shape = RoundedCornerShape(16.dp),
47                 elevation = CardDefaults.cardElevation(8.dp),
48                 modifier = Modifier
49                     .fillMaxWidth()
50                     .padding(vertical = 8.dp)
51             ) {
52                 Row(
53                     modifier = Modifier
54                         .padding(16.dp)

```

```

54         .fillMaxWidth()
55     ) {
56         GlideImage(
57             imageUrl = item.imageUrl,
58             contentDescription = item.Name,
59             modifier = Modifier
60                 .height(150.dp)
61                 .width(100.dp)
62                 .clip(RoundedCornerShape(12.dp)),
63             contentScale = ContentScale.Crop
64         )
65
66         Spacer(modifier = Modifier.width(16.dp))
67
68         Column(
69             modifier = Modifier.weight(1f)
70         ) {
71             Row(
72                 modifier
73                 Modifier.fillMaxWidth(),
74                 horizontalArrangement
75                 Arrangement.SpaceBetween
76             ) {
77                 Text(
78                     text = item.Name,
79                     style
80                     MaterialTheme.typography.titleMedium
81                 )
82                 Text(
83                     text = item.HumanHost,
84                     style
85                     MaterialTheme.typography.bodyMedium
86                 )
87             }
88
89             Spacer(modifier
90                 Modifier.height(4.dp))
91
92             Row(
93                 modifier
94                 Modifier.fillMaxWidth(),
95                 horizontalArrangement
96                 Arrangement.SpaceBetween
97             ) {
98                 Text(
99                     text = "Info: ",
100                     style
101                     MaterialTheme.typography.bodySmall,
102                     fontWeight = FontWeight.Bold
103                 )
104                 Text(
105                     text = item.Detail,

```

98	style	=
99	MaterialTheme.typography.bodySmall,	
100	maxLines = 4,	=
	overflow	
	TextOverflow.Ellipsis	
101	)	
102	}	
103		
104	Spacer(modifier	=
	Modifier.height(8.dp))	
105		
106	Row(	
107	horizontalArrangement	=
	Arrangement.spacedBy(30.dp)	
108	) {	
109	Button(	
110	onClick = {	
111	val intent	=
	Intent(Intent.ACTION_VIEW, item.link.toUri())	
112	navController.context.startActivity(intent)	
113	},	
114	modifier	=
	Modifier.padding(start = 30.dp)	
115	) {	
116	Text("Wiki")	
117	}	
118		
119	Button(	
120	onClick = {	
121	navController.navigate("detail/\${item.id}")	
122	}	
123	) {	
124	Text("Detail")	
125	}	
126	}	
127	}	
128	}	
129	}	
130	}	
131	}	
132	}	

Tabel 4. Source Code Jawaban Soal 1

## 5. DetailScreen.kt

1	package com.example.ultraman.ui.screens
2	
3	import android.widget.ImageView

```

4 import androidx.compose.foundation.layout.Column
5 import androidx.compose.foundation.layout.Spacer
6 import androidx.compose.foundation.layout.WindowInsets
7 import androidx.compose.foundation.layout.asPaddingValues
8 import androidx.compose.foundation.layout.fillMaxSize
9 import androidx.compose.foundation.layout.fillMaxWidth
10 import androidx.compose.foundation.layout.height
11 import androidx.compose.foundation.layout.padding
12 import androidx.compose.foundation.layout.systemBars
13 import androidx.compose.foundation.rememberScrollState
14 import androidx.compose.foundation.shape.RoundedCornerShape
15 import androidx.compose.foundation.verticalScroll
16 import androidx.compose.material3.MaterialTheme
17 import androidx.compose.material3.Text
18 import androidx.compose.runtime.Composable
19 import androidx.compose.ui.Modifier
20 import androidx.compose.ui.draw.clip
21 import androidx.compose.ui.layout.ContentScale
22 import androidx.compose.ui.platform.LocalContext
23 import androidx.compose.ui.text.font.FontWeight
24 import androidx.compose.ui.text.style.TextAlign
25 import androidx.compose.ui.unit.dp
26 import androidx.compose.ui.viewinterop.AndroidView
27 import com.bumptech.glide.Glide
28 import com.example.ultraman.ultramans
29
30 @Composable
31 fun DetailScreen(itemId: Int?) {
32     val item = ultramans.find { it.id == itemId }
33     item?.let {
34         Column(
35             modifier = Modifier
36                 .fillMaxSize()
37                 .padding(16.dp)
38
39             .padding(WindowInsets.systemBars.asPaddingValues())
40                 .verticalScroll(rememberScrollState())
41         ) {
42             GlideImage(
43                 imageUrl = it.imageUrl,
44                 contentDescription = it.Name,
45                 modifier = Modifier
46                     .fillMaxWidth()
47                     .height(600.dp)
48                     .clip(RoundedCornerShape(16.dp)),
49                 contentScale = ContentScale.Crop
50             )
51             Spacer(modifier = Modifier.height(12.dp))
52             Text(it.Name,
53                 style = MaterialTheme.typography.headlineSmall,
54                 fontWeight = FontWeight.Bold)
55         }
56     }
57 }

```

```

52         Text( text = "Human Host: ", fontWeight =
FontWeight.Bold)
53         Text(it.HumanHost)
54         Text( text = "Height: ", fontWeight =
FontWeight.Bold)
55         Text(it.Height)
56         Text( text = "Weight: ", fontWeight =
FontWeight.Bold)
57         Text(it.Weight)
58         Text( text = "\nUltraman Info: ", fontWeight =
FontWeight.Bold)
59         Text( text = it.Detail, textAlign =
TextAlign.Justify)
60     }
61 }
62 }
63
64 @Composable
65 fun GlideImage(
66     imageUrl: String,
67     contentDescription: String?,
68     modifier: Modifier = Modifier,
69     contentScale: ContentScale = ContentScale.Crop
70 ) {
71     val context = LocalContext.current
72     AndroidView(
73         factory = {
74             ImageView(context).apply {
75                 scaleType = when (contentScale) {
76                     ContentScale.Crop ->
ImageView.ScaleType.CENTER_CROP
77                     ContentScale.Fit ->
ImageView.ScaleType.FIT_CENTER
78                     else -> ImageView.ScaleType.CENTER_CROP
79                 }
80                 contentDescription?.let {
81                     this.contentDescription = it }
82             },
83         update = {
84             Glide.with(context)
85                 .load(imageUrl)
86                 .into(it)
87         },
88         modifier = modifier
89     )
90 }

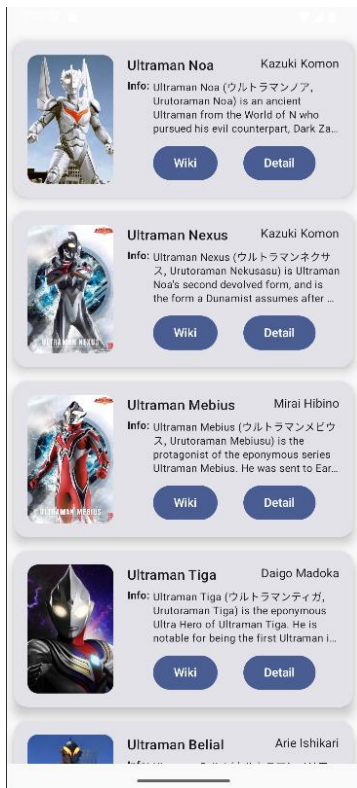
```

Tabel 5. Source Code Jawaban Soal 1



## B. Output Program

- Tampilan halaman list ultraman :



Gambar 3. Screenshot Hasil Jawaban Soal 1

- Tampilan Halaman detail ultraman :



Gambar 4. Screenshot Hasil Jawaban Soal 1



Gambar 5. Screenshot Hasil Jawaban Soal 1

## C. Pembahasan

### 1. MainActivity.kt

Pada bagian awal, saya mendeklarasikan package `com.example.ultraman`, yang menjadi namespace dari file ini. Kemudian saya mengimpor berbagai class dan fungsi yang diperlukan, seperti `ComponentActivity` untuk activity utama, `setContent` untuk menampilkan UI berbasis Jetpack Compose, serta fungsi navigasi seperti `NavHost`, `composable`, dan `rememberNavController`. Saya juga mengimpor komponen buatan sendiri seperti `ListScreen`, `DetailScreen`, dan tema aplikasi `UltramanTheme`.

Kelas `MainActivity` adalah titik masuk utama aplikasi. Saya menjadikannya turunan dari `ComponentActivity` karena saya menggunakan Jetpack Compose sebagai kerangka kerja UI. Di dalam fungsi `onCreate()`, saya pertama-tama memanggil `super.onCreate()` untuk menjalankan logika bawaan Android. Setelah itu, saya menggunakan `WindowCompat.setDecorFitsSystemWindows(window, false)` supaya tampilan bisa tampil penuh sampai ke tepi layar (edge-to-edge), melewati batas status bar atau navigation bar.

Selanjutnya, saya memanggil `setContent` untuk mulai menampilkan UI dengan Compose. Di dalamnya, saya bungkus seluruh tampilan dengan `UltramanTheme`, agar seluruh elemen UI mengikuti aturan warna dan tipografi yang konsisten. Saya kemudian membuat sebuah `NavHostController` dengan `rememberNavController()`, yang akan saya gunakan untuk mengelola navigasi antar layar. Terakhir, saya panggil fungsi `AppNavHost` dan memberikan `navController` sebagai argumennya untuk mengatur semua rute di aplikasi ini.

Fungsi `AppNavHost` adalah fungsi `@Composable` yang saya buat untuk mendefinisikan sistem navigasi dalam aplikasi. Saya menggunakan `NavHost` untuk mengatur daftar rute, dengan `startDestination` ditetapkan ke `"list"`, artinya layar pertama yang muncul adalah daftar Ultraman. Di dalam `NavHost`, saya mendefinisikan rute pertama yaitu `"list"`, yang akan menampilkan `ListScreen` dan meneruskan `navController` agar layar tersebut bisa melakukan navigasi lebih lanjut.

Rute kedua yang saya definisikan adalah `"detail/{itemId}"`. Rute ini memiliki parameter `itemId` bertipe integer, yang akan dikirimkan saat pengguna memilih salah satu item di daftar. Di dalam blok `composable` tersebut, saya mengambil nilai `itemId` dari `backStackEntry.arguments`, lalu meneruskannya ke `DetailScreen` agar layar detail bisa menampilkan informasi sesuai dengan item yang dipilih. Dengan cara ini, aplikasi saya memiliki navigasi dua layar: daftar dan detail, yang terhubung dinamis lewat parameter.

### 2. ListUltraman.kt

`ListUltraman.kt` ini fungsinya adalah sebagai bagian dari struktur data atau model dalam aplikasi Ultraman saya. Di dalamnya, saya mendefinisikan sebuah data class bernama `ListUltraman`. Data class ini berfungsi sebagai blueprint atau cetakan data untuk setiap karakter Ultraman yang akan saya tampilkan di aplikasi, baik dalam bentuk daftar maupun detail.

Data class `ListUltraman` memiliki delapan properti utama. Pertama, ada `id` bertipe `Int` sebagai penanda unik untuk setiap Ultraman. Kemudian, `Name` menyimpan nama Ultramanya, dan `HumanHost` menyimpan nama host manusianya jika ada. Dua properti berikutnya adalah `Height` dan `Weight`, yang menyimpan tinggi dan berat tubuh Ultraman dalam bentuk `String`, karena biasanya data ini datang dalam satuan teks seperti `"49 m"` atau `"40,000 t"`.

Lalu ada properti Detail, yang berisi deskripsi atau penjelasan lengkap tentang Ultraman tersebut, serta imageUrl, yaitu link gambar yang akan saya tampilkan di aplikasi. Terakhir, saya juga menambahkan properti link, yang berisi URL ke halaman Wikipedia atau sumber lain jika pengguna ingin membaca lebih lanjut.

### **3. DataUltraman.kt**

Di awal file, saya mengimpor ListUltraman dari package model, karena saya akan membuat daftar karakter Ultraman yang disusun menggunakan data class tersebut. Model ini sebelumnya sudah saya siapkan dengan properti seperti id, Name, HumanHost, Height, Weight, Detail, imageUrl, dan link.

Selanjutnya, saya mendeklarasikan variabel ultramans sebagai val, yang berarti nilainya tetap dan tidak bisa diubah. Variabel ini merupakan listOf dari objek ListUltraman, jadi isinya adalah daftar karakter Ultraman lengkap beserta semua data penting yang diperlukan untuk ditampilkan di UI, seperti nama, host manusianya, tinggi dan berat, deskripsi, gambar, serta link Wikipedia untuk informasi lebih lanjut.

Setiap item dalam list merepresentasikan satu karakter Ultraman. Misalnya, karakter pertama adalah Ultraman Noa dengan host bernama Kazuki Komon. Deskripsinya saya ambil dari sumber referensi fandom resmi agar informasinya akurat dan lengkap. Saya juga menyertakan URL gambar yang akan ditampilkan dalam aplikasi dan tautan ke halaman wiki untuk tiap Ultraman.

Data ini nantinya akan digunakan oleh tampilan ListScreen untuk menampilkan daftar karakter, dan saat salah satu dipilih, itemId dari masing-masing objek akan dikirim ke halaman DetailScreen untuk menampilkan informasi lengkap.

### **4. ListScreen.kt**

Kode ini merupakan tampilan utama daftar Ultraman dalam bentuk composable bernama ListScreen, yang menerima parameter NavController untuk navigasi antar layar. Saya menggunakan LazyColumn untuk menampilkan daftar secara scrollable secara efisien. Modifier seperti fillMaxSize, padding, dan WindowInsets.systemBars.asPaddingValues() saya gunakan agar kontennya memenuhi layar dengan padding yang menyesuaikan status bar atau sistem UI.

Di dalam LazyColumn, saya memanggil items(ultramans) untuk menampilkan setiap objek ListUltraman dalam bentuk Card. Setiap card memiliki bentuk rounded dan elevasi agar tampil seperti kotak mengambang. Konten dalam card diatur dalam Row horizontal, di mana bagian kiri menampilkan gambar Ultraman menggunakan GlideImage dengan ukuran tetap dan sudut membulat, sementara bagian kanan berisi informasi teks yang ditampilkan secara vertikal dalam Column.

Bagian informasi terdiri dari nama Ultraman (item.Name) dan nama host manusianya (item.HumanHost) yang disusun dalam Row agar sejajar. Lalu ada deskripsi singkat (item.Detail) yang dibatasi maksimal 4 baris dan diberi TextOverflow.Ellipsis agar tidak memanjang keluar batas. Setelah itu, terdapat dua tombol: tombol "Wiki" yang akan membuka link ke halaman Fandom dengan menggunakan Intent dan fungsi toUri(), serta

tombol "Detail" yang akan menavigasi ke halaman detail dengan `navController.navigate("detail/${item.id}").`

## **5. DetailScreen.kt**

Fungsi `DetailScreen` menampilkan informasi lengkap tentang satu karakter Ultraman berdasarkan `itemId` yang diterima sebagai parameter. Data diambil dari list ultramans dengan fungsi `find`. Jika data ditemukan, maka kontennya ditampilkan dalam sebuah `Column` yang dapat di-scroll secara vertikal. Seluruh isi layar diberi padding termasuk dari sistem UI (status bar, navigation bar) agar tampil rapi dan responsif.

Gambar Ultraman ditampilkan menggunakan fungsi `GlideImage`, yaitu wrapper untuk `AndroidView` yang menampilkan `ImageView` dan memuat gambar menggunakan `Glide`. Gambar ditampilkan dengan lebar penuh, tinggi 600dp, dan bentuk membulat di sudutnya. Setelah gambar, ditampilkan teks-teks berisi nama Ultraman, human host-nya, tinggi dan berat badan, serta deskripsi atau detail tambahan yang diratakan ke kiri-kanan (`TextAlign.Justify`).

Sementara itu, fungsi `GlideImage` sendiri menerima URL gambar, deskripsi, dan modifier. Ia menggunakan `AndroidView` untuk membuat dan mengupdate `ImageView`, lalu memuat gambar dari internet menggunakan `Glide`. Skala tampilan gambar disesuaikan berdasarkan nilai `ContentScale` dari `Compose`.

## **Mengapa RecyclerView masih digunakan, dibandingkan LazyColumn dengan kode yang lebih singkat?**

### **1. Proyek Legacy (warisan)**

Banyak aplikasi Android lama (yang dibangun sebelum `Compose` rilis stabil) masih menggunakan XML + `RecyclerView`. Mengganti seluruh arsitektur ke `Compose` bisa mahal dan berisiko, jadi `RecyclerView` tetap dipertahankan.

### **2. Kustomisasi Ekstrem & Kompatibilitas**

`RecyclerView` memiliki ekosistem yang luas: seperti `ItemTouchHelper`, `ConcatAdapter`, `Paging`, dan dukungan untuk berbagai layout manager (`GridLayoutManager`, `StaggeredGridLayoutManager`). Beberapa fitur atau fleksibilitas ini belum sepenuhnya setara di `LazyColumn`.

### **3. Performansi pada kasus tertentu**

Meski `LazyColumn` cukup efisien, `RecyclerView` masih unggul dalam pengelolaan memory dan recycling yang sangat detail di beberapa kasus edge (misalnya list dengan ribuan item kompleks), karena sudah sangat matang dan teruji.

### **4. Kompatibilitas dengan View**

Jika proyek masih menggunakan fragment dan layout XML, RecyclerView lebih cocok karena Compose tidak bisa disisipkan dengan fleksibilitas penuh tanpa ComposeView wrapper, yang justru menambah kompleksitas baru.

### **5. Penggunaan di Library atau SDK**

Beberapa library UI (misalnya di Maps SDK, Ads SDK) atau komponen pihak ketiga masih berbasis View system dan RecyclerView, sehingga lebih mudah integrasi jika tetap pakai View-based list.

### **D. Tautan Git**

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/Omelette719/Pemrograman-Mobile.git>