LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Muhammad Azwin Hakim **NIM. 2310817310012**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

Laporan Akhir Praktikum Pemrograman Mobile ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Akhir Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Azwin Hakim
NIM : 2310817310012

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

DAFTAR ISI

DAFTAR GAMBAR

MODUL 1 : ANDROID BASIC WITH KOTLIN

MODUL 2 : ANDROID LAYOUT

MODUL 3 : BUILD A SCROLLABLE LIST

MODUL 4 : VIEWMODEL AND DEBUGGING

MODUL 5 : CONNECT TO THE INTERNET

DAFTAR TABEL

MODUL 1 : ANDROID BASIC WITH KOTLIN

MODUL 2 : ANDROID LAYOUT

MODUL 3 : BUILD A SCROLLABLE LIST

MODUL 4 : VIEWMODEL AND DEBUGGING

MODUL 5 : CONNECT TO THE INTERNET

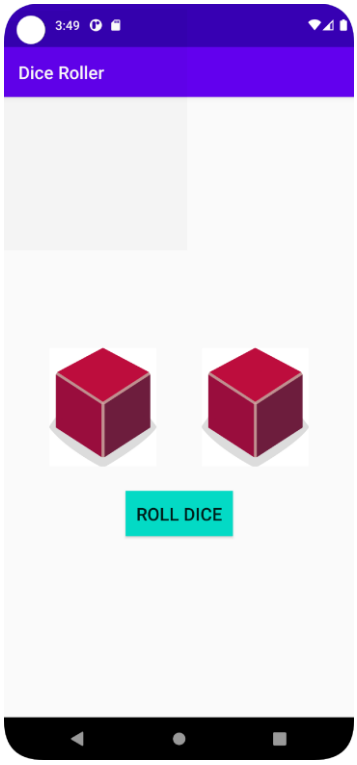
MODUL 1 :
ANDROID BASIC WITH KOTLIN

SOAL 1

Soal Praktikum:

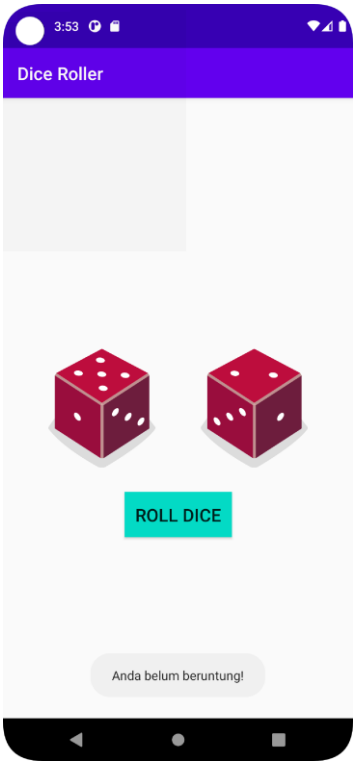
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



Gambar 1. Tampilan Awal Aplikasi Modul 1

2. Setelah user menekan tombol “Roll Dice” maka masing-masing dadu akan memunculkan sisi dadu masing-masing dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2 maka akan menampilkan pesan “Anda belum beruntung!” seperti dapat dilihat pada Gambar 2.

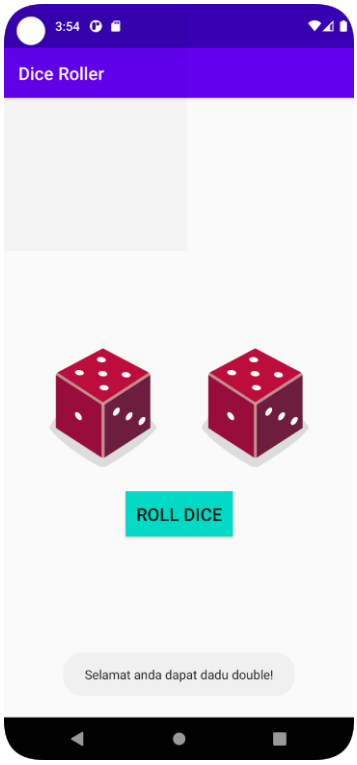


Gambar 2. Tampilan Dadu Setelah Di Roll Modul 1

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan “Selamat anda dapat dadu double!” seperti dapat dilihat pada Gambar 3.

4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.

5. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export= download



Gambar 3. Tampilan Roll Dadu Double Modul 1

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1 Modul 1

```
1  package com.example.diceroller
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.activity.enableEdgeToEdge
6  import androidx.appcompat.app.AppCompatActivity
7  import androidx.core.view.ViewCompat
8  import androidx.core.view.WindowInsetsCompat
9  import com.example.diceroller.databinding.ActivityMainBinding
10
11 class MainActivity : AppCompatActivity() {
```

```
12
13      private lateinit var binding: ActivityMainBinding
14
15      private var diceRoll = 0
16      private var diceRollTwo = 0
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          enableEdgeToEdge()
21
22          binding = ActivityMainBinding.inflate(layoutInflater)
23          setContentView(binding.root)
24
25          ViewCompat.setOnApplyWindowInsetsListener(binding.main)
    { v, insets ->
26              val                    systemBars                    =
    insets.getInsets(WindowInsetsCompat.Type.systemBars())
27              v.setPadding(systemBars.left,       systemBars.top,
    systemBars.right, systemBars.bottom)
28              insets
29          }
30
31          binding.button.setOnClickListener { rollDice() }
32
33          if (savedInstanceState != null) {
34              diceRoll = savedInstanceState.getInt("diceRoll", 0)
35              diceRollTwo                                           =
    savedInstanceState.getInt("diceRollTwo", 0)
36          }
37
38          updateDiceImages()
39      }
40
41      override fun onSaveInstanceState(outState: Bundle) {
```

```
42          super.onSaveInstanceState(outState)
43          outState.putInt("diceRoll", diceRoll)
44          outState.putInt("diceRollTwo", diceRollTwo)
45      }
46
47      private fun rollDice() {
48          val dice = Dice(6)
49          val diceTwo = Dice(6)
50
51          diceRoll = dice.roll()
52          diceRollTwo = diceTwo.roll()
53
54          updateDiceImages()
55
56          if (diceRoll == diceRollTwo) {
57              Toast.makeText(this,  "Selamat  anda  dapat  dadu
    double!", Toast.LENGTH_SHORT).show()
58          } else {
59              Toast.makeText(this,  "Anda  belum  beruntung!",
    Toast.LENGTH_SHORT).show()
60          }
61      }
62
63      private fun updateDiceImages() {

64  binding.imageView.setImageResource(getDiceImage(diceRoll))
            binding.imageView.contentDescription                =
65  diceRoll.toString()



66  binding.imageView2.setImageResource(getDiceImage(diceRollTwo))
67          binding.imageView2.contentDescription               =
68  diceRollTwo.toString()
    }
```

```
69
70      private fun getDiceImage(value: Int): Int {
71          return when (value) {
72              1 -> R.drawable.dice_1
73              2 -> R.drawable.dice_2
74              3 -> R.drawable.dice_3
75              4 -> R.drawable.dice_4
76              5 -> R.drawable.dice_5
77              6 -> R.drawable.dice_6
78              else -> R.drawable.dice_0
79          }
80      }
81  }
82
83  class Dice(private val numSides: Int) {
84      fun roll(): Int = (1..numSides).random()
85  }
86
```

2. activity_main.xml

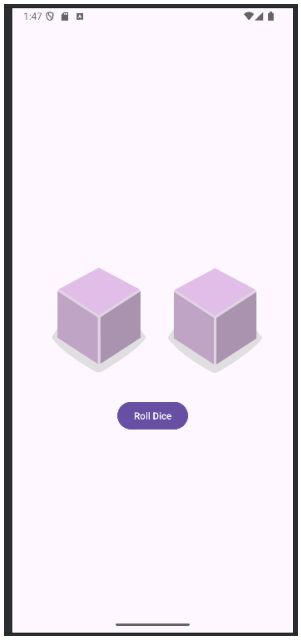
Tabel 2. Source Code Jawaban Soal 1 Modul 1

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

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="Roll Dice"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView" />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="160dp"
        android:layout_height="200dp"
        android:contentDescription="@android:string/no"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.187"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.499"
        tools:srcCompat="@drawable/dice_0" />

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="160dp"
        android:layout_height="200dp"
        android:contentDescription="@android:string/no"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.864"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@drawable/dice_0" />

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

B. Output Program

- Tampilan awal :

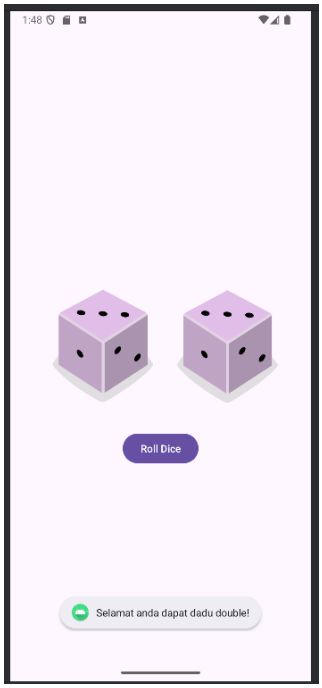


Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1

- Tampilan jika nilai dadu berbeda :

Gambar 5. Screenshot Hasil Jawaban Soal 1 Modul 1

- Tampilan jika dadu bernilai sama :



Gambar 6. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

1. MainActivity.kt:

Kode pertama adalah file Kotlin MainActivity.kt yang berfungsi sebagai logika utama aplikasi Android untuk melempar dua buah dadu. Aktivitas ini menggunakan ViewBinding, seperti yang ditunjukkan oleh deklarasi "private lateinit var binding: ActivityMainBinding", untuk menghubungkan elemen-elemen di XML dengan kode Kotlin tanpa perlu menggunakan findViewById.

Pada metode onCreate, aplikasi mengatur layout dengan "setContentView(binding.root)", lalu menetapkan listener pada tombol "binding.button.setOnClickListener { rollDice() }" untuk menjalankan fungsi rollDice() saat tombol ditekan. Fungsi rollDice() sendiri menggunakan class Dice yang didefinisikan di bagian bawah kode ("class Dice(private val numSides: Int)") untuk menghasilkan angka acak antara 1 hingga 6 dengan "diceRoll = dice.roll()" dan "diceRollTwo = diceTwo.roll()". Setelah angka acak didapatkan, fungsi updateDiceImages() dipanggil untuk memperbarui gambar dadu sesuai angka yang muncul, menggunakan fungsi getDiceImage(value: Int), yang mengembalikan ID resource drawable berdasarkan nilai dadu. Selain itu, jika kedua dadu menunjukkan angka yang sama ("if (diceRoll == diceRollTwo)"), maka akan muncul toast "Selamat anda dapat dadu double!", jika tidak, toast "Anda belum beruntung!" akan ditampilkan. Untuk menangani rotasi layar, onSaveInstanceState digunakan untuk menyimpan nilai diceRoll dan diceRollTwo agar dapat dipulihkan saat aktivitas dibuat ulang.

2. activity_main.xml

Kode kedua adalah file XML activity_main.xml yang mendefinisikan tampilan antarmuka pengguna menggunakan ConstraintLayout sebagai root layout. Di dalam layout ini terdapat sebuah tombol dengan ID "@+id/button" yang berfungsi sebagai pemicu untuk melempar dadu. Tombol ini ditempatkan secara horizontal di tengah dan vertikal di bawah ImageView pertama melalui atribut app:layout_constraintTop_toBottomOf="@+id/imageView". Dua ImageView digunakan untuk menampilkan gambar dua buah dadu, dengan ID masing-masing "@+id/imageView" dan "@+id/imageView2". Kedua gambar ini ditempatkan secara

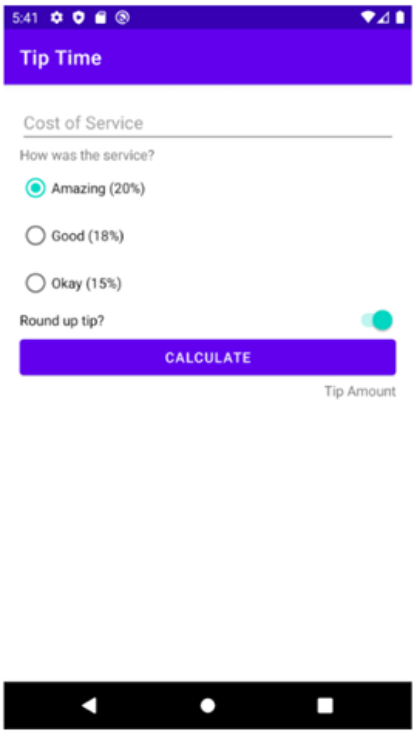
horizontal di sisi kiri dan kanan layout, yang dikontrol melalui atribut app:layout_constraintHorizontal_bias. Gambar awal yang digunakan untuk kedua dadu adalah @drawable/dice_0, yang merupakan gambar dadu kosong atau posisi awal sebelum dadu dilempar. Layout ini memiliki atribut tools:context=".MainActivity" yang menunjukkan bahwa layout terkait dengan MainActivity.

MODUL 2 :
ANDROID LAYOUT

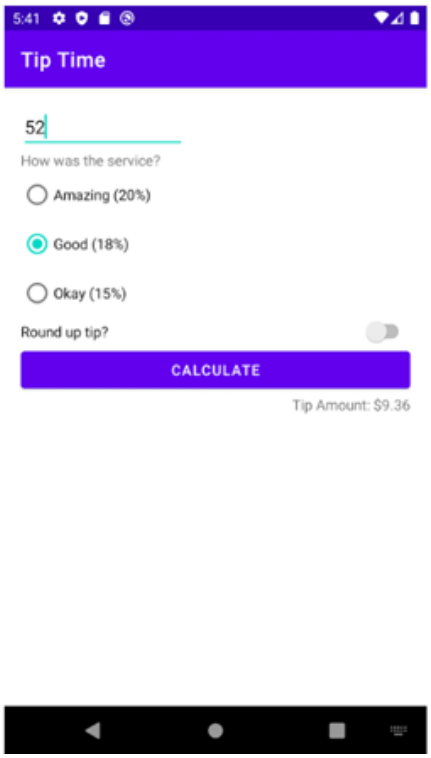
SOAL 1

Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

6. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.
7. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.
8. Pengaturan Pembulatan Tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.
9. Tampilan Hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 7. Tampilan Awal Aplikasi Modul 2



Gambar 8. Tampilan Aplikasi Setelah Dijalankan Modul 2

A. Source Code

1. MainActivity.kt

Tabel 3. Source Code Jawaban Soal 1 Modul 2

```
1   package com.example.tiptime
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.ComponentActivity
6   import androidx.activity.compose.setContent
7   import androidx.compose.foundation.layout.*
8   import androidx.compose.foundation.text.KeyboardActions
9   import androidx.compose.foundation.text.KeyboardOptions
10  import androidx.compose.material3.*
11  import androidx.compose.runtime.*
12  import androidx.compose.ui.Alignment
13  import androidx.compose.ui.Modifier
14  import androidx.compose.ui.platform.LocalContext
15  import androidx.compose.ui.platform.LocalFocusManager
16  import androidx.compose.ui.text.input.ImeAction
17  import androidx.compose.ui.text.input.KeyboardType
18  import androidx.compose.ui.unit.dp
19  import androidx.compose.foundation.rememberScrollState
20  import androidx.compose.foundation.verticalScroll
21  import com.example.tiptime.ui.theme.TipTimeTheme
22  import androidx.compose.runtime.saveable.rememberSaveable
23  import kotlin.math.ceil
24
25  class MainActivity : ComponentActivity() {
26      override fun onCreate(savedInstanceState: Bundle?) {
27          super.onCreate(savedInstanceState)
28          setContent {
29              TipTimeTheme {
30                  Surface(
```

```
                    modifier = Modifier.fillMaxSize(),
                    color                                    =
MaterialTheme.colorScheme.background
                ) {
                    TipTimeScreen()
                }
            }
        }
    }
}

@Composable
fun TipTimeScreen() {
    var amountInput by rememberSaveable { mutableStateOf("") }
    var     tipPercentage     by     rememberSaveable     {
mutableDoubleStateOf(0.20) }
    var roundUp by rememberSaveable { mutableStateOf(false) }
    var tipResult by rememberSaveable { mutableStateOf("") }

    val context = LocalContext.current
    val focusManager = LocalFocusManager.current
    val scrollState = rememberScrollState()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)
            .verticalScroll(scrollState)
    ) {
        OutlinedTextField(
            value = amountInput,
            onValueChange = { amountInput = it },
            label = { Text("Cost of Service") },
            singleLine = true,
```

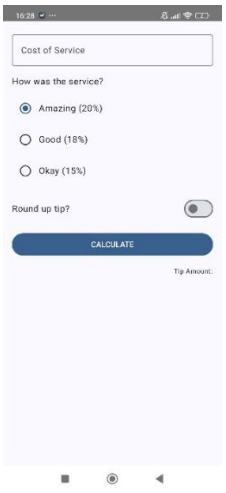
```
63              keyboardOptions = KeyboardOptions.Default.copy(
64                  keyboardType = KeyboardType.Number,
65                  imeAction = ImeAction.Done
66              ),
67              keyboardActions = KeyboardActions(
68                  onDone = { focusManager.clearFocus() }
69              ),
70              modifier = Modifier.fillMaxWidth()
71          )
72
73          Spacer(modifier = Modifier.height(16.dp))
74
75          Text("How was the service?")
76          Spacer(modifier = Modifier.height(8.dp))
77
78          val radioOptions = listOf(
79              "Amazing (20%)" to 0.20,
80              "Good (18%)" to 0.18,
81              "Okay (15%)" to 0.15
82          )
83
84          radioOptions.forEach { (label, value) ->
85              Row(
86                  verticalAlignment                                =
    Alignment.CenterVertically,
87                  modifier = Modifier.padding(vertical = 4.dp)
88              ) {
89                  RadioButton(
90                      selected = tipPercentage == value,
91                      onClick = { tipPercentage = value }
92                  )
93                  Text(label)
94              }
95          }
```

```
96
97          Spacer(modifier = Modifier.height(16.dp))
98
99          Row(
100             verticalAlignment = Alignment.CenterVertically,
101             horizontalArrangement = Arrangement.SpaceBetween,
102             modifier = Modifier.fillMaxWidth()
103         ) {
104             Text("Round up tip?")
105             Switch(
106                 checked = roundUp,
107                 onCheckedChange = { roundUp = it }
108             )
109         }
110
111         Spacer(modifier = Modifier.height(16.dp))
112
113         Button(
114             onClick = {
115                 val amount = amountInput.toDoubleOrNull()
116
117                 if (amount == null || amount <= 0) {
118                     Toast.makeText(context, "Harus di isi dan
    berupa angka!", Toast.LENGTH_SHORT).show()
119                 } else {
120                     var tip = amount * tipPercentage
121                     if (roundUp) {
122                         tip = ceil(tip)
123                     }
124                     tipResult = "Tip Amount: $%.2f".format(tip)
125                 }
126             },
127             modifier = Modifier.fillMaxWidth()
128         ) {
```

```
129                 Text("CALCULATE")
130             }
131
132             Spacer(modifier = Modifier.height(16.dp))
133
134             Box(
135                 modifier = Modifier.fillMaxWidth()
136             ) {
137                 Text(
138                     text = if (tipResult.isEmpty()) "Tip Amount:"
    else tipResult,
139                     style = MaterialTheme.typography.bodySmall,
140                     modifier = Modifier.align(Alignment.CenterEnd)
141                 )
142             }
143         }
144 }
```

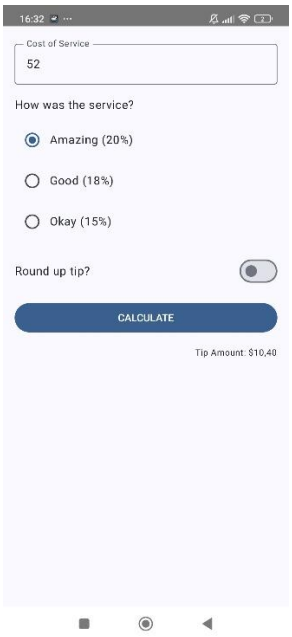
B. Output Program

- Tampilan awal :

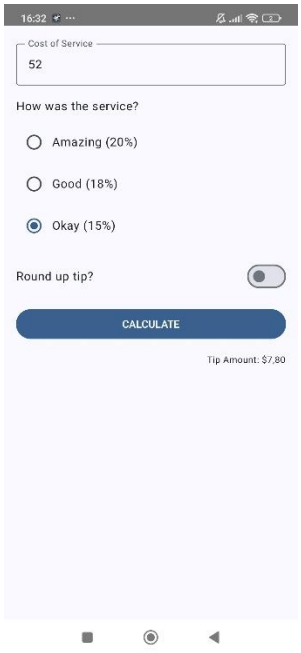


Gambar 9. Screenshot Hasil Jawaban Soal 1 Modul 2

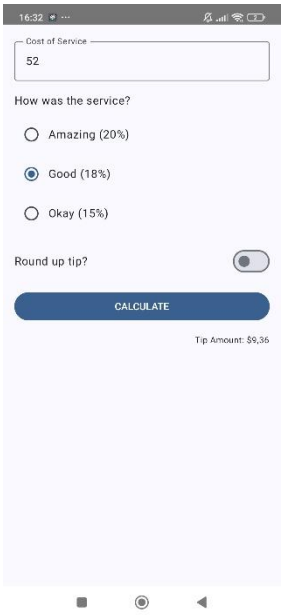
- Tampilan Aplikasi Setelah Dijalankan :



Gambar 10. Screenshot Hasil Jawaban Soal 1 Modul 2

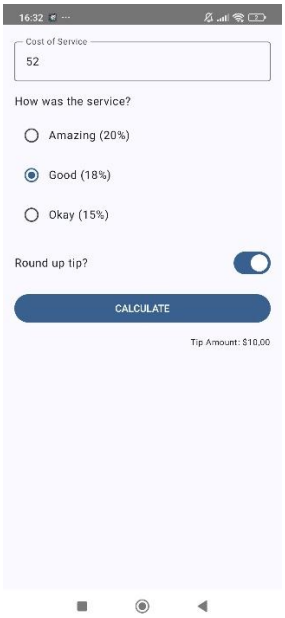


Gambar 11. Screenshot Hasil Jawaban Soal 1 Modul 2



Gambar 12. Screenshot Hasil Jawaban Soal 1 Modul 2

- Hasil Pembulatan Tip :



Gambar 13. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

Kode dimulai dengan mendeklarasikan kelas MainActivity, yang merupakan kelas utama yang mengelola tampilan aplikasi. Pada metode onCreate, aplikasi memanggil fungsi setContent untuk menetapkan antarmuka pengguna menggunakan Jetpack Compose. Di dalam setContent, tema aplikasi TipTimeTheme diterapkan, dan tampilan utama dari aplikasi ini adalah hasil dari pemanggilan fungsi komposisi TipTimeScreen. Fungsi ini berisi seluruh logika tampilan dan interaksi pengguna.

Di dalam fungsi TipTimeScreen, terdapat beberapa variabel yang menyimpan data input dan status pengguna. Variabel amountInput digunakan untuk menyimpan nilai yang dimasukkan pengguna sebagai biaya layanan. Variabel tipPercentage menyimpan persentase tip yang dipilih oleh pengguna, seperti 20% untuk layanan yang luar biasa, 18% untuk layanan yang baik, dan 15% untuk layanan yang cukup. Variabel roundUp adalah sebuah boolean yang menunjukkan apakah pengguna ingin membulatkan jumlah tip ke angka terdekat. Hasil perhitungan tip disimpan dalam variabel tipResult, yang akan menampilkan hasil akhir perhitungan tip di layar. Semua variabel ini disimpan dengan menggunakan rememberSaveable, yang memungkinkan data ini tetap bertahan meskipun aplikasi dihentikan atau digulir.

Antarmuka pengguna terdiri dari beberapa elemen UI yang terstruktur dalam kolom. OutlinedTextField digunakan untuk menerima input biaya layanan dari pengguna, di mana hanya angka yang dapat dimasukkan. Keyboard yang muncul saat pengguna mengetik dikonfigurasi untuk hanya menerima angka, dan ada opsi untuk menekan tombol "Done" yang menghapus fokus dari input. Selain itu, ada tiga opsi radio button yang memungkinkan pengguna untuk memilih persentase tip yang sesuai dengan pengalaman layanan mereka, yaitu 20%, 18%, atau 15%. Opsi lain yang disediakan adalah Switch, yang memungkinkan pengguna memilih apakah mereka ingin membulatkan jumlah tip yang dihitung.

Ketika pengguna menekan tombol "CALCULATE", aplikasi memeriksa apakah input biaya layanan yang dimasukkan valid. Jika nilai yang dimasukkan tidak valid,

aplikasi akan menampilkan pesan toast yang mengingatkan pengguna untuk memasukkan angka yang valid. Jika input valid, aplikasi kemudian menghitung jumlah tip berdasarkan biaya layanan yang dimasukkan dan persentase yang dipilih. Jika opsi "Round up" diaktifkan, jumlah tip dibulatkan menggunakan fungsi ceil untuk memastikan angka tersebut dibulatkan ke atas. Hasil perhitungan kemudian ditampilkan pada layar, dengan format yang menunjukkan jumlah tip yang perlu diberikan.

Aplikasi ini juga menyediakan mekanisme untuk mengelola fokus pengguna dengan menggunakan LocalFocusManager, yang memungkinkan untuk menghapus fokus dari elemen input setelah pengguna selesai mengisinya. Selain itu, tampilan aplikasi dapat digulirkan menggunakan rememberScrollState dan verticalScroll, yang berguna ketika keyboard muncul dan mengambil ruang layar.

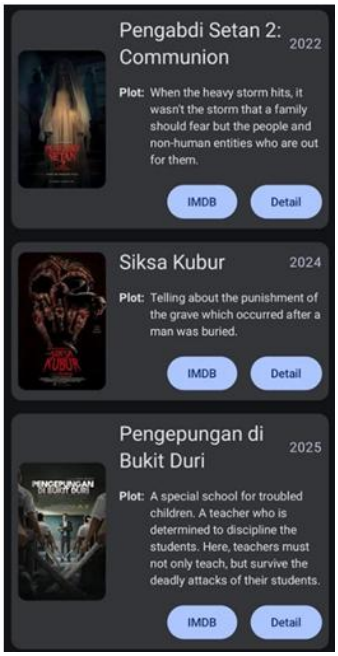
MODUL 3 :
BUILD A SCROLLABLE LIST

SOAL 1

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

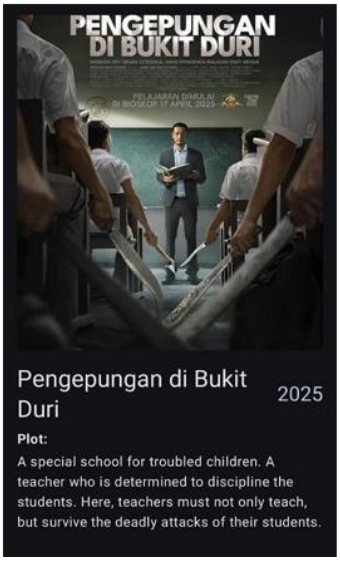
1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
    a. Button pertama 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

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 14. Contoh UI List Modul 3

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 15. Contoh UI Detail Modul 3

A. Source Code

1. MainActivity.kt

Tabel 4. Source Code Jawaban Soal 1 Modul 3

```
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
```

```
@Composable
fun AppNavHost(navController: NavHostController) {
    NavHost(navController = navController, startDestination = "list") {
        composable("list") {
            ListScreen(navController)
        }
        composable(
            "detail/{itemId}",
            arguments = listOf(navArgument("itemId") { type = NavType.IntType })
        ) { backStackEntry ->
            val itemId = backStackEntry.arguments?.getInt("itemId")
            DetailScreen(itemId)
        }
    }
}
```

2. ListUltraman.kt

Tabel 5. Source Code Jawaban Soal 1 Modul 3

```
package com.example.ultraman.model

data class ListUltraman(
    val id: Int,
    val Name: String,
    val HumanHost: String,
    val Height: String,
    val Weight: String,
    val Detail: String,
    val imageUrl: String,
    val link: String
)
```

3. DataUltraman.kt

Tabel 6. Source Code Jawaban Soal 1 Modul 3

```
1   package com.example.ultraman
2
3   import com.example.ultraman.model.ListUltraman
4
5   val ultramans = listOf(
6       ListUltraman(
            id = 1,
            name = "Ultraman Noa",
            host = "Kazuki Komon",
            height = "55 m",
            weight = "55,000 t",
            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.",
            imageUrl                                              =
    "https://th.bing.com/th/id/OIP.1rd_goNDK6agE1vRe4AMyAHaKX?rs=
    1&pid=ImgDetMain",
            wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Noa"
        ),
7       ListUltraman(
            id = 2,
            name = "Ultraman Nexus",
            host = "Kazuki Komon",
            height = "49 m",
            weight = "40,000 t",
```

```
        description = "Ultraman Nexus (ウルトラマンネクサス, Urutoraman Nekusasu) is Ultraman Noa's second devolved form, 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.",
        imageUrl = "https://pbs.twimg.com/media/FVhSSBHagAAWSju.jpg",
        wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Nexus_(character)"
    ),
8   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)"
    ),
9   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)"
    ),
    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" +
11      "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"),
12          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" +
13      "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&pid=ImgDetMain",
        "https://ultra.fandom.com/wiki/Ultraman_Zero")
14      )
```

4. ListScreen.kt

Tabel 7. Source Code Jawaban Soal 1 Modul 3

```
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
    .padding(WindowInsets.systemBars.asPaddingValues())
42      ) {
43          items(ultramans) { item ->
44              Card(
45                  shape = RoundedCornerShape(16.dp),
46                  elevation = CardDefaults.cardElevation(8.dp),
47                  modifier = Modifier
48                      .fillMaxWidth()
49                      .padding(vertical = 8.dp)
50              ) {
51                  Row(
52                      modifier = Modifier
53                          .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                        =
    Modifier.fillMaxWidth(),
73                              horizontalArrangement           =
    Arrangement.SpaceBetween
74                          ) {
75                              Text(
76                                  text = item.Name,
77                                  style                       =
    MaterialTheme.typography.titleMedium
78                              )
79                              Text(
80                                  text = item.HumanHost,
81                                  style                       =
    MaterialTheme.typography.bodyMedium
82                              )
83                          }
84
85                          Spacer(modifier                     =
    Modifier.height(4.dp))
86
87                          Row(
88                              modifier                        =
    Modifier.fillMaxWidth(),
89                              horizontalArrangement           =
    Arrangement.SpaceBetween
90                          ) {
91                              Text(
92                                  text = "Info: ",
```

```
93                                  style                           =
    MaterialTheme.typography.bodySmall,
94                                  fontWeight = FontWeight.Bold
95                              )
96                              Text(
97                                  text = item.Detail,
98                                  style                           =
    MaterialTheme.typography.bodySmall,
99                                  maxLines = 4,
100                                 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 }
```

5. DetailScreen.kt

Tabel 8. Source Code Jawaban Soal 1 Modul 3

```
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
```

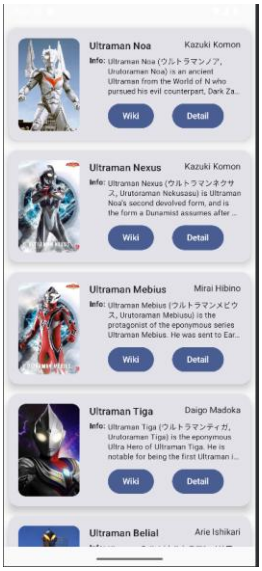
```
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.viewinterop.AndroidView
import com.bumptech.glide.Glide
import com.example.ultraman.ultramans

@Composable
fun DetailScreen(itemId: Int?) {
    val item = ultramans.find { it.id == itemId }
    item?.let {
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(16.dp)

.padding(WindowInsets.systemBars.asPaddingValues())
                .verticalScroll(rememberScrollState())
        ) {
            GlideImage(
                imageUrl = it.imageUrl,
                contentDescription = it.Name,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(600.dp)
                    .clip(RoundedCornerShape(16.dp)),
```

```
48                  contentScale = ContentScale.Crop
49              )
50              Spacer(modifier = Modifier.height(12.dp))
51              Text(it.Name,                style                =
    MaterialTheme.typography.headlineSmall,       fontWeight       =
    FontWeight.Bold)
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                          {
    this.contentDescription = it }
81              }
82          },
83          update = {
84              Glide.with(context)
85                  .load(imageUrl)
86                  .into(it)
87          },
88          modifier = modifier
89      )
90  }
```

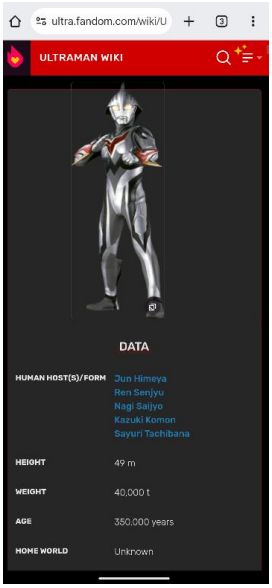
B. Output Program

- Tampilan halaman list ultraman :



Gambar 16. Screenshot Hasil Jawaban Soal 1 Modul 3

- Ketika Button Wiki di klik :



Gambar 17. Screenshot Hasil Jawaban Soal 1 Modul 3

- Tampilan Halaman detail ultraman :

Gambar 18. Screenshot Hasil Jawaban Soal 1 Modul 3

Gambar 19. Screenshot Hasil Jawaban Soal 1 Modul 3

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 Ultraman-nya, 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 NavHostController 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.

SOAL 2

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

A. Pembahasan

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.

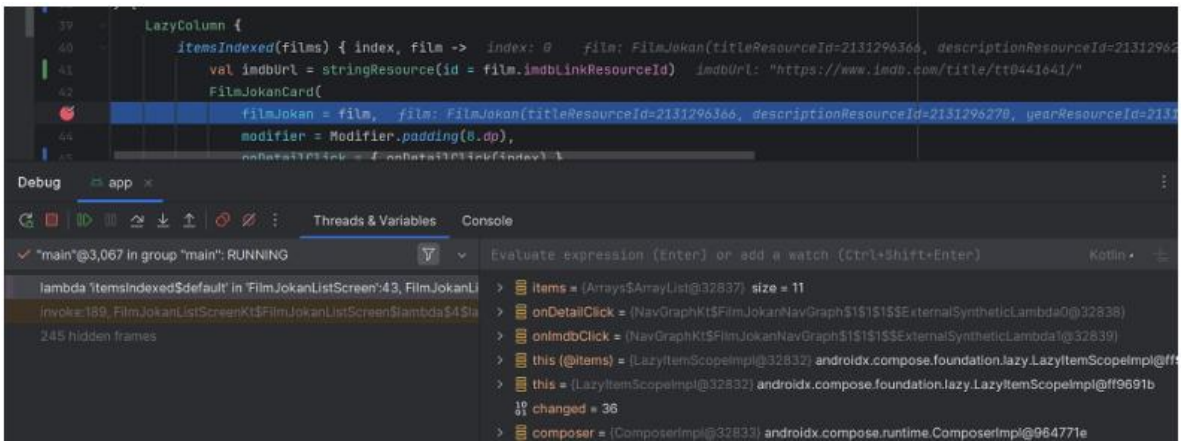
MODUL 4 :
VIEWMODEL AND DEBUGGING

SOAL 1

Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:

a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
b. Gunakan ViewModelFactory dalam pembuatan ViewModel
c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
d. gunakan logging untuk event berikut:
    a. Log saat data item masuk ke dalam list
    b. Log saat tombol Detail dan tombol Explicit Intent ditekan
    c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

Aplikasi harus dapat mempertahankan fitur-fitur yang sudah dibuat pada modul sebelumnya. Berikut adalah contoh debugging dalam Android Studio.



Gambar 20. Contoh Penggunaan Debugger Modul 4

A. Source Code

1. MainActivity.kt

Tabel 9. Source Code Jawaban Soal 1 Modul 4

```
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.compose.runtime.Composable
7  import androidx.lifecycle.viewmodel.compose.viewModel
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.ViewModel.UltramanViewModel
15 import
   com.example.ultraman.ui.ViewModel.UltramanViewModelFactory
16 import com.example.ultraman.ui.screens.DetailScreen
17 import com.example.ultraman.ui.screens.ListScreen
18 import com.example.ultraman.ui.theme.UltramanTheme
```

```
19
20 class MainActivity : ComponentActivity() {
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContent {
24             UltramanTheme {
25                 val navController = rememberNavController()
26                 val viewModel: UltramanViewModel =
   viewModel(factory = UltramanViewModelFactory())
27                 AppNavHost(navController, viewModel)
28             }
29         }
30     }
31 }
32
33 @Composable
34 fun AppNavHost(navController: NavHostController, viewModel:
   UltramanViewModel) {
35     NavHost(navController = navController, startDestination =
   "list") {
36         composable("list") {
37             ListScreen(navController, viewModel)
38         }
39         composable(
40             "detail/{itemId}",
41             arguments = listOf(navArgument("itemId") { type =
   NavType.IntType })
42         ) { backStackEntry ->
43             val itemId =
   backStackEntry.arguments?.getInt("itemId")
44             DetailScreen(itemId, viewModel)
45         }
46     }
47 }
```

2. ListUltraman.kt

Tabel 10. Source Code Jawaban Soal 1 Modul 4

```
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  )
```

3. DataUltraman.kt

Tabel 11. Source Code Jawaban Soal 1 Modul 4

```
1   package com.example.ultraman
2
3   import com.example.ultraman.model.ListUltraman
4
5   val ultramans = listOf(
6       ListUltraman(
            id = 1,
            name = "Ultraman Noa",
            host = "Kazuki Komon",
            height = "55 m",
            weight = "55,000 t",
            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.",
        imageUrl = "https://th.bing.com/th/id/OIP.1rd_goNDK6agE1vRe4AMyAHaKX?rs=1&pid=ImgDetMain",
        wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Noa"
    ),
7   ListUltraman(
        id = 2,
        name = "Ultraman Nexus",
        host = "Kazuki Komon",
        height = "49 m",
        weight = "40,000 t",
        description = "Ultraman Nexus (ウルトラマンネクサス, Urutoraman Nekusasu) is Ultraman Noa's second devolved form, 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.",
        imageUrl = "https://pbs.twimg.com/media/FVhSSBHagAAWSju.jpg",
        wikiUrl = "https://ultra.fandom.com/wiki/Ultraman_Nexus_(character)"
    ),
8   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)"
    ),
9   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)"
```

```
    ),
10  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" +
11  "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"),
12  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" +
        "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&pid=ImgDetMain",
        "https://ultra.fandom.com/wiki/Ultraman_Zero")
)
```

4. UltramanViewModel.kt

Tabel 12. Source Code Jawaban Soal 1 Modul 4

```
package com.example.ultraman.ui.ViewModel

import android.util.Log
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.example.ultraman.model.ListUltraman
import com.example.ultraman.ultramans
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow

class UltramanViewModel : ViewModel() {

    private val _ultramanList = MutableStateFlow<List<ListUltraman>>(emptyList())
    val ultramanList: StateFlow<List<ListUltraman>> = _ultramanList

    init {
```

```
17          _ultramanList.value = ultramans
18          Log.d("UltramanViewModel", "Data list dimasukkan ke
    ViewModel (${ultramans.size} item)")
19      }
20
21      fun getItemById(id: Int): ListUltraman? {
22          return _ultramanList.value.find { it.id == id }
23      }
24  }
25
26  class UltramanViewModelFactory : ViewModelProvider.Factory {
27      @Suppress("UNCHECKED_CAST")
28      override fun <T : ViewModel> create(modelClass: Class<T>):
    T {
29          return UltramanViewModel() as T
30      }
31  }
```

5. ListScreen.kt

Tabel 13. Source Code Jawaban Soal 1 Modul 4

```
1   package com.example.ultraman.ui.screens
2
3   import android.content.Intent
4   import android.util.Log
5   import androidx.compose.foundation.layout.Arrangement
6   import androidx.compose.foundation.layout.Column
7   import androidx.compose.foundation.layout.Row
8   import androidx.compose.foundation.layout.Spacer
9   import androidx.compose.foundation.layout.WindowInsets
10  import androidx.compose.foundation.layout.fillMaxSize
11  import androidx.compose.foundation.layout.fillMaxWidth
12  import androidx.compose.foundation.layout.height
13  import androidx.compose.foundation.layout.padding
```

```
14  import androidx.compose.foundation.layout.systemBars
15  import androidx.compose.foundation.layout.asPaddingValues
16  import androidx.compose.ui.text.style.TextOverflow
17  import androidx.compose.foundation.layout.width
18  import androidx.compose.foundation.lazy.LazyColumn
19  import androidx.compose.foundation.shape.RoundedCornerShape
20  import androidx.compose.material3.Button
21  import androidx.compose.material3.Card
22  import androidx.compose.material3.CardDefaults
23  import androidx.compose.material3.MaterialTheme
24  import androidx.compose.material3.Text
25  import androidx.compose.runtime.Composable
26  import androidx.compose.ui.Modifier
27  import androidx.compose.ui.draw.clip
28  import androidx.compose.ui.layout.ContentScale
29  import androidx.compose.ui.unit.dp
30  import androidx.navigation.NavHostController
31  import androidx.compose.foundation.lazy.items
32  import androidx.compose.runtime.collectAsState
33  import androidx.compose.ui.text.font.FontWeight
34  import androidx.core.net.toUri
35  import com.example.ultraman.ui.ViewModel.UltramanViewModel
36
37  @Composable
38  fun ListScreen(navController: NavHostController, viewModel:
    UltramanViewModel) {
39      val ultramanList =
    viewModel.ultramanList.collectAsState().value
40
41      LazyColumn(
42          modifier = Modifier
43              .fillMaxSize()
44              .padding(8.dp)
```

```
45
    .padding(WindowInsets.systemBars.asPaddingValues())
46      ) {
47          items(ultramanList) { item ->
48              Card(
49                  shape = RoundedCornerShape(16.dp),
50                  elevation = CardDefaults.cardElevation(8.dp),
51                  modifier = Modifier
52                      .fillMaxWidth()
53                      .padding(vertical = 8.dp)
54              ) {
55                  Row(
56                      modifier = Modifier
57                          .padding(16.dp)
58                          .fillMaxWidth()
59                  ) {
60                      GlideImage(
61                          imageUrl = item.imageUrl,
62                          contentDescription = item.Name,
63                          modifier = Modifier
64                              .height(150.dp)
65                              .width(100.dp)
66                              .clip(RoundedCornerShape(12.dp)),
67                          contentScale = ContentScale.Crop
68                      )
69
70                      Spacer(modifier = Modifier.width(16.dp))
71
72                      Column(
73                          modifier = Modifier.weight(1f)
74                      ) {
75                          Row(
76                              modifier                            =
    Modifier.fillMaxWidth(),
```

```
77                              horizontalArrangement              =
    Arrangement.SpaceBetween
78                          ) {
79                              Text(
80                                  text = item.Name,
81                                  style                          =
    MaterialTheme.typography.titleMedium
82                              )
83                              Text(
84                                  text = item.HumanHost,
85                                  style                          =
    MaterialTheme.typography.bodyMedium
86                              )
87                          }
88
89                          Spacer(modifier                        =
    Modifier.height(4.dp))
90
91                          Row(
92                              modifier                           =
    Modifier.fillMaxWidth(),
93                              horizontalArrangement              =
    Arrangement.SpaceBetween
94                          ) {
95                              Text(
96                                  text = "Info: ",
97                                  style                          =
    MaterialTheme.typography.bodySmall,
98                                  fontWeight = FontWeight.Bold
99                              )
100                             Text(
101                                 text = item.Detail,
102                                 style                          =
    MaterialTheme.typography.bodySmall,
```

```
103                                 maxLines = 4,
104                                 overflow                        =
    TextOverflow.Ellipsis
105                             )
106                         }
107
108                         Spacer(modifier                         =
    Modifier.height(8.dp))
109
110                         Row(
111                             horizontalArrangement               =
    Arrangement.spacedBy(30.dp)
112                         ) {
113                             Button(
114                                 onClick = {
115                                     Log.d("ListScreen",
    "Tombol Wiki ditekan untuk ${item.Name}")
116                                 val        intent        =
    Intent(Intent.ACTION_VIEW, item.link.toUri())
117
    navController.context.startActivity(intent)
118                                 }
119                             ) {
120                                 Text("Wiki")
121                             }
122
123                             Button(
124                                 onClick = {
125                                     Log.d("ListScreen",
    "Tombol Detail ditekan untuk ${item.Name}")
126
    navController.navigate("detail/${item.id}")
127                                 }
128                             ) {
```

```
129                                     Text("Detail")
130                                 }
131                             }
132                         }
133                     }
134                 }
135             }
136         }
137 }
```

6. DetailScreen.kt

Tabel 14. Source Code Jawaban Soal 1 Modul 4

```
1  package com.example.ultraman.ui.screens
2
3  import android.util.Log
4  import android.widget.ImageView
5  import androidx.compose.foundation.layout.Column
6  import androidx.compose.foundation.layout.Spacer
7  import androidx.compose.foundation.layout.WindowInsets
8  import androidx.compose.foundation.layout.asPaddingValues
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.rememberScrollState
15 import androidx.compose.foundation.shape.RoundedCornerShape
16 import androidx.compose.foundation.verticalScroll
17 import androidx.compose.material3.MaterialTheme
18 import androidx.compose.material3.Text
19 import androidx.compose.runtime.Composable
20 import androidx.compose.ui.Modifier
21 import androidx.compose.ui.draw.clip
```

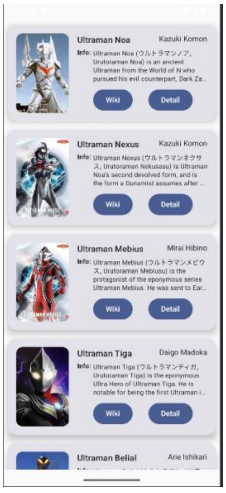
```
22 import androidx.compose.ui.layout.ContentScale
23 import androidx.compose.ui.platform.LocalContext
24 import androidx.compose.ui.text.font.FontWeight
25 import androidx.compose.ui.text.style.TextAlign
26 import androidx.compose.ui.unit.dp
27 import androidx.compose.ui.viewinterop.AndroidView
28 import com.bumptech.glide.Glide
29 import com.example.ultraman.ui.ViewModel.UltramanViewModel
30
31 @Composable
32 fun DetailScreen(itemId: Int?, viewModel: UltramanViewModel) {
33     val item = viewModel.getItemById(itemId ?: -1)
34     Log.d("DetailScreen", "Navigasi ke DetailScreen untuk ID:
   $itemId - ${item?.Name}")
35
36     item?.let {
37         Column(
38             modifier = Modifier
39                 .fillMaxSize()
40                 .padding(16.dp)
41
   .padding(WindowInsets.systemBars.asPaddingValues())
42                 .verticalScroll(rememberScrollState())
43         ) {
44             GlideImage(
45                 imageUrl = it.imageUrl,
46                 contentDescription = it.Name,
47                 modifier = Modifier
48                     .fillMaxWidth()
49                     .height(600.dp)
50                     .clip(RoundedCornerShape(16.dp)),
51                 contentScale = ContentScale.Crop
52             )
53             Spacer(modifier = Modifier.height(12.dp))
```

```
54              Text(it.Name,                  style                 =
    MaterialTheme.typography.headlineSmall,       fontWeight       =
    FontWeight.Bold)
55              Text(  text  =  "Human  Host:  ",  fontWeight  =
    FontWeight.Bold)
56              Text(it.HumanHost)
57              Text(   text   =   "Height:   ",   fontWeight   =
    FontWeight.Bold)
58              Text(it.Height)
59              Text(   text   =   "Weight:   ",   fontWeight   =
    FontWeight.Bold)
60              Text(it.Weight)
61              Text(  text  =  "\nUltraman  Info:  ",  fontWeight  =
    FontWeight.Bold)
62              Text(    text    =    it.Detail,    textAlign    =
    TextAlign.Justify)
63          }
64      }
65  }
66
67  @Composable
68  fun GlideImage(
69      imageUrl: String,
70      contentDescription: String?,
71      modifier: Modifier = Modifier,
72      contentScale: ContentScale = ContentScale.Crop
73  ) {
74      val context = LocalContext.current
75      AndroidView(
76          factory = {
77              ImageView(context).apply {
78                  scaleType = when (contentScale) {
79                      ContentScale.Crop                          ->
    ImageView.ScaleType.CENTER_CROP
```

```
80                      ContentScale.Fit                                ->
    ImageView.ScaleType.FIT_CENTER
81                      else -> ImageView.ScaleType.CENTER_CROP
82                  }
83                  contentDescription?.let                             {
    this.contentDescription = it }
84              }
85          },
86          update = {
87              Glide.with(context)
88                  .load(imageUrl)
89                  .into(it)
90          },
91          modifier = modifier
92      )
93  }
```

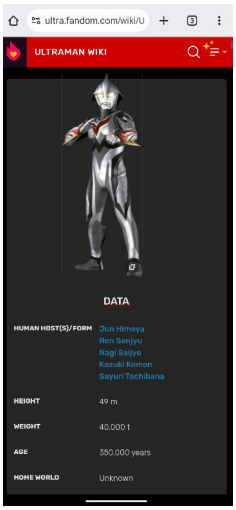
B. Output Program

- Tampilan halaman list ultraman :



Gambar 21. Screenshot Hasil Jawaban Soal 1 Modul 4

- Ketika Button Wiki di klik :



Gambar 22. Screenshot Hasil Jawaban Soal 1 Modul 4

- Tampilan Halaman detail ultraman :

Gambar 23. Screenshot Hasil Jawaban Soal 1 Modul 4

Gambar 24. Screenshot Hasil Jawaban Soal 1 Modul 4

C. Pembahasan

Pembahasan Kode :

1. MainActivity.kt

Di dalam onCreate, saya memanggil setContent untuk mengatur isi UI menggunakan UltramanTheme agar tampilannya konsisten dengan tema aplikasi. Lalu saya membuat instance NavHostController dengan rememberNavController() untuk mengatur navigasi. Saya juga membuat UltramanViewModel dengan viewModel(factory = UltramanViewModelFactory()) agar data bisa dipisahkan dari UI dan mudah dikelola.

Navigasinya saya definisikan di fungsi AppNavHost, dengan dua rute utama:

- "list" untuk menampilkan daftar Ultraman melalui ListScreen.
- "detail/{itemId}" untuk menampilkan detail tiap Ultraman yang dipilih. Argumen itemId saya ambil dari back stack dan dikirim ke DetailScreen untuk menampilkan informasi lengkap berdasarkan ID-nya.

Struktur ini sengaja saya buat agar navigasi antar layar tetap sederhana tapi fleksibel, sekaligus mengikuti arsitektur MVVM yang rapi.

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 Ultraman-nya, 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. UltramanViewModel.kt

Kode ini saya buat sebagai ViewModel untuk menyimpan dan mengelola data Ultraman secara terpisah dari UI. Tujuannya adalah agar data tetap hidup meskipun terjadi rotasi layar atau perubahan konfigurasi lainnya, dan untuk mempermudah observasi data di Jetpack Compose.

Pada bagian atas, saya deklarasikan _ultramanList sebagai MutableStateFlow, lalu saya expose versi read-only-nya lewat val ultramanList. Ini berguna supaya composable bisa mengamati data tapi tidak bisa mengubahnya langsung, sesuai prinsip encapsulation.

Di dalam blok init, saya langsung isi _ultramanList dengan data dummy ultramans, yang berasal dari file statis. Saya juga menambahkan log agar saat debugging, saya tahu kapan dan berapa banyak data yang berhasil dimuat ke dalam ViewModel.

Saya juga buat fungsi getItemById(id: Int) agar bisa mengambil satu item berdasarkan id. Ini sangat berguna ketika navigasi ke halaman detail karena hanya perlu mengambil satu objek dari list, bukan mengoper semua data.

Terakhir, saya buat UltramanViewModelFactory, karena saat inject ViewModel ke dalam Compose, saya perlu factory custom untuk membuat instance-nya. Dengan cara ini, ViewModel saya bisa dipakai dengan viewModel(factory = ...).

5. ListScreen.kt

Kode ini saya buat untuk menampilkan daftar karakter Ultraman di halaman utama aplikasi, menggunakan LazyColumn agar performanya tetap efisien meskipun datanya banyak. Setiap item dalam daftar ditampilkan dalam bentuk Card, yang di dalamnya berisi gambar, nama Ultraman, nama Human Host, deskripsi singkat, serta dua tombol aksi: Wiki dan Detail.

Saya ambil datanya dari UltramanViewModel menggunakan collectAsState() agar data tersebut bisa direaktifkan dan Compose bisa otomatis melakukan re-composition saat data berubah.Gambarnya saya tampilkan dengan GlideImage, yang saya buat sendiri

menggunakan AndroidView dan Glide, karena Glide belum punya dukungan Compose secara langsung.

Teks "Info" saya buat dengan gaya lebih tebal menggunakan fontWeight = FontWeight.Bold, sementara detailnya dibatasi maksimal 4 baris agar tidak memakan terlalu banyak ruang dan diberi Ellipsis jika kepanjangan. Ini menjaga UI tetap rapi.

Untuk tombol Wiki, saya pakai Intent.ACTION_VIEW untuk membuka tautan eksternal (biasanya halaman Wikipedia atau sumber lain) di browser. Untuk tombol Detail, saya arahkan ke halaman detail menggunakan navController.navigate("detail/${item.id}"), dan ID tersebut nanti akan digunakan untuk mengambil data spesifik di halaman detail.

Saya juga menambahkan baris log menggunakan Log.d() sebagai alat bantu debugging yang sederhana namun efektif. Log ini dicetak saat pengguna menekan tombol "Wiki" atau "Detail" pada setiap item Ultraman. Untuk tombol "Wiki", log mencatat pesan bahwa tombol tersebut ditekan, lengkap dengan nama karakter Ultraman yang sedang dibuka, sehingga saya bisa memantau apakah intent menuju halaman Wikipedia berhasil dipicu. Begitu juga dengan tombol "Detail", saya mencatat interaksi pengguna saat mereka ingin melihat detail lebih lanjut tentang karakter tersebut. Log semacam ini sangat berguna saat pengembangan aplikasi karena memberikan gambaran langsung di logcat tentang bagaimana pengguna berinteraksi dengan UI. Dari log, saya bisa tahu apakah tombol-tombol berfungsi sesuai harapan dan dapat segera melacak jika terjadi kesalahan atau navigasi yang tidak berjalan sebagaimana mestinya.

6. DetailScreen.kt

Dalam kode DetailScreen, saya menggunakan Log.d() sebagai alat bantu untuk mencatat aktivitas navigasi ke layar detail. Log ini mencetak ID dari item yang dipilih, dan jika item berhasil ditemukan dari ViewModel, maka nama karakter Ultraman-nya juga ikut dicetak. Hal ini sangat membantu saya untuk memastikan bahwa data yang dikirim dari halaman sebelumnya (melalui navigasi) benar-benar sampai dan diproses dengan baik. Bila suatu saat halaman ini tidak menampilkan data, saya bisa langsung melihat di logcat apakah nilai itemId kosong, tidak sesuai, atau jika getItemById() gagal menemukan datanya. Dengan kata lain, log ini menjadi bukti bahwa proses pengambilan data sudah berjalan — atau tempat pertama yang saya periksa jika ada bug.

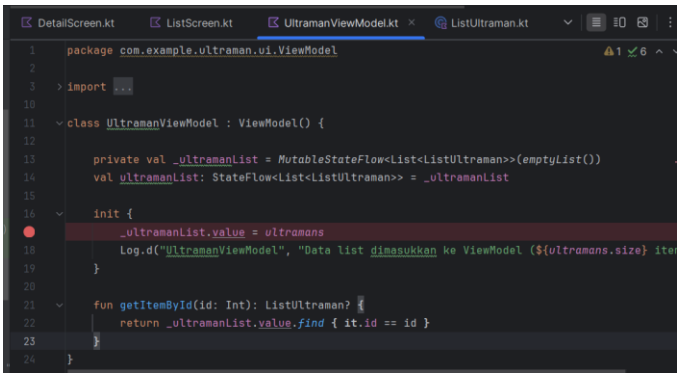
Selain itu, layar ini memanfaatkan komponen GlideImage untuk memuat gambar dari URL. Komponen ini menggunakan AndroidView agar bisa memakai ImageView klasik dan Glide, karena Coil memang belum saya gunakan di proyek ini. Ini memungkinkan gambar tetap dimuat dengan efisien meskipun saya menggunakan Jetpack Compose sebagai framework UI utama. Saya juga sengaja menggunakan verticalScroll untuk memastikan bahwa seluruh detail dari karakter Ultraman bisa terlihat meskipun panjang, dan styling teks saya atur agar rapi, dengan bagian deskripsi rata kiri-kanan menggunakan TextAlign.Justify untuk kenyamanan membaca.

Penjelasan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out

Debugger merupakan alat penting dalam pengembangan aplikasi Android yang digunakan untuk mendeteksi, melacak, dan memperbaiki kesalahan logika dalam kode program saat aplikasi berjalan. Tidak seperti log biasa yang hanya menampilkan hasil keluaran, debugger memberikan kontrol penuh kepada developer untuk menghentikan jalannya aplikasi di titik tertentu, memeriksa kondisi variabel, serta memahami alur eksekusi secara real-time. Dengan begitu, debugger memungkinkan kita untuk menemukan bug yang tidak terlihat hanya dari hasil log.

Cara Menggunakan Debugger :

1. Pasang Breakpoint:
    - Klik di sebelah kiri baris kode (gutter) untuk menambahkan titik merah (breakpoint).



Gambar 25. Screenshot Contoh Menambahkan Breakpoint pada kode Modul 4

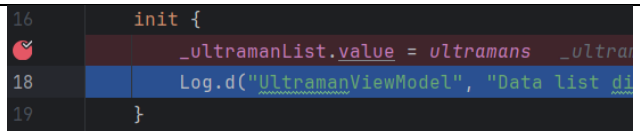
2. Jalankan dengan Mode Debug:
    - Tekan tombol Debug di toolbar Android Studio, disamping tombol run.
3. Ketika Breakpoint Tercapai:
    - Aplikasi akan pause di baris tersebut.
4. Menggunakan Step Into, Step Over, dan Step Out

Setelah aplikasi berhenti pada breakpoint, kamu bisa mulai menelusuri alur kode:

- Step Over (F8): Menjalankan baris kode saat ini dan langsung lompat ke baris berikutnya, tanpa masuk ke fungsi yang dipanggil.

Cocok digunakan di baris seperti ini:

```
Log.d("UltramanViewModel", ...)
```

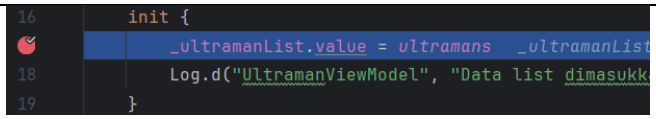


Gambar 26. Screenshot Contoh Step Over Kebaris Selanjutnya Modul 4

- Step Into (F7): Masuk ke dalam fungsi yang dipanggil di baris tersebut, jika kamu ingin menelusuri implementasi fungsi tersebut.

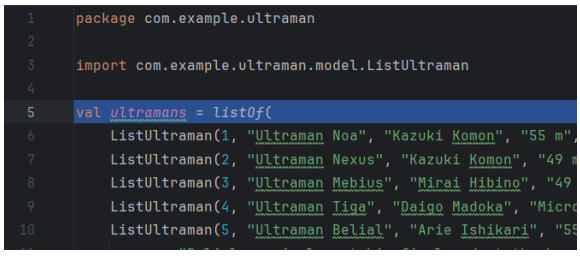
Misalnya :

```
_ultramanList.value = ultramans
```



Gambar 27. Screenshot Contoh Step On Kebaris ini Modul 4

Jika ultramans adalah properti atau fungsi, kamu bisa menelusuri ke dalamnya.



Gambar 28. Screenshot Contoh Bagian Dalam Fungsi Modul 4

- Step Out (Shift + F8): Digunakan jika kamu sudah masuk terlalu dalam ke suatu fungsi dan ingin langsung keluar ke fungsi pemanggil sebelumnya.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Application class dalam arsitektur aplikasi Android dan fungsinya

Dalam arsitektur aplikasi Android, kelas Application berperan sebagai titik awal dan pusat dari siklus hidup aplikasi. Kelas ini merupakan turunan dari android.app.Application dan hanya dibuat sekali oleh sistem Android, sebelum Activity, Service, atau komponen lainnya dijalankan. Karena bersifat global dan memiliki rentang hidup sepanjang aplikasi berjalan, Application sering dimanfaatkan untuk menginisialisasi komponen-komponen penting yang digunakan secara luas di seluruh aplikasi.

Salah satu penggunaan utama kelas Application adalah untuk mengatur dependency injection, seperti Dagger atau Hilt, serta konfigurasi awal untuk Retrofit, Room, atau library lain yang dibutuhkan oleh berbagai bagian aplikasi. Selain itu, kelas ini juga kerap digunakan untuk menyimpan status global, seperti informasi pengguna yang sedang login, meskipun penggunaan ini harus dibatasi agar tidak menyebabkan memory leak.

Di luar itu, kelas Application juga menjadi tempat yang ideal untuk menginisialisasi library pihak ketiga, seperti Firebase, Crashlytics, Timber, atau layanan analitik lainnya. Bahkan, kamu bisa menambahkan logika khusus seperti mendeteksi apakah aplikasi sedang berada di foreground atau background, serta menjalankan observer untuk kondisi jaringan. Dengan kata lain, Application merupakan fondasi tempat berbagai inisialisasi dan konfigurasi awal dilakukan sebelum komponen UI dimulai.

MODUL 5 :
CONNECT TO THE INTERNET

SOAL 1

Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi

sesuai ketentuan berikut:

a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
b. Gunakan KotlinX Serialization sebagai library JSON.
c. Gunakan library seperti Coil atau Glide untuk image loading.
d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API:

https://developer.themoviedb.org/docs/getting-started

e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
f. Gunakan caching strategy pada Room..
g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya.

A. Source Code

1. MainActivity.kt

Tabel 15. Source Code Jawaban Soal 1 Modul 5

```
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.compose.foundation.layout.padding
7  import androidx.compose.material3.Scaffold
8  import androidx.compose.runtime.Composable
9  import androidx.compose.ui.Modifier
10 import androidx.lifecycle.viewmodel.compose.viewModel
11 import androidx.navigation.NavHostController
12 import androidx.navigation.NavType
13 import androidx.navigation.compose.*
14 import androidx.navigation.navArgument
15 import com.example.ultraman.ui.ViewModel.UltramanViewModel
16 import
   com.example.ultraman.ui.ViewModel.UltramanViewModelFactory
17 import com.example.ultraman.ui.components.BottomBarNavigation
18 import com.example.ultraman.ui.screens.DetailScreen
19 import com.example.ultraman.ui.screens.FavoriteScreen
20 import com.example.ultraman.ui.screens.ListScreen
21 import com.example.ultraman.ui.theme.UltramanTheme
22
23 class MainActivity : ComponentActivity() {
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContent {
27             UltramanTheme {
28                 val navController = rememberNavController()
29                 val viewModel: UltramanViewModel = viewModel(
```

```
                    factory =
UltramanViewModelFactory(applicationContext)
                )
                UltramanApp(navController, viewModel)
            }
        }
    }
}

@Composable
fun UltramanApp(
    navController: NavHostController,
    viewModel: UltramanViewModel
) {
    Scaffold(
        bottomBar = {
            BottomBarNavigation(navController = navController)
        }
    ) { innerPadding ->
        NavHost(
            navController = navController,
            startDestination = "list",
            modifier = Modifier.padding(innerPadding)
        ) {
            composable("list") {
                ListScreen(navController, viewModel)
            }
            composable("favorite") {
                FavoriteScreen(navController, viewModel)
            }

            composable(
                "detail/{itemId}",
```

```
62                 arguments = listOf(navArgument("itemId") { type
   = NavType.IntType })
63             ) { backStackEntry ->
64                 val                    itemId                    =
   backStackEntry.arguments?.getInt("itemId")
65                 DetailScreen(itemId, viewModel)
66             }
67         }
68     }
69 }
```

2. Model/UltramanItem.kt

Tabel 16. Source Code Jawaban Soal 1 Modul 5

```
1  package com.example.ultraman.model
2
3  import androidx.room.Entity
4  import androidx.room.PrimaryKey
5  import kotlinx.serialization.SerialName
6  import kotlinx.serialization.Serializable
7
8  @Serializable
9  @Entity(tableName = "ultraman")
10 data class UltramanItem(
11     @PrimaryKey val id: Int,
12     val title: String,
13
14     @SerialName("overview")
15     val description: String,
16
17     @SerialName("poster_path")
18     val posterPath: String? = null,
19
20     @SerialName("release_date")
```

```
    val releaseDate: String? = null,

    @SerialName("first_air_date")
    val firstAirDate: String? = null,

    val isFavorite: Boolean = false
) {
    val year: String
        get() = releaseDate?.take(4) ?: firstAirDate?.take(4) ?:
"????"

    fun getPosterUrl(): String? = posterPath?.let {
        "https://image.tmdb.org/t/p/w500$it"
    }

    fun getTmdbLink(): String {
        return if (!releaseDate.isNullOrBlank()) {
            "https://www.themoviedb.org/movie/$id"
        } else {
            "https://www.themoviedb.org/tv/$id"
        }
    }
}
```

3. Model/SearchResponse.kt

Tabel 17. Source Code Jawaban Soal 1 Modul 5

```
package com.example.ultraman.model

import kotlinx.serialization.SerialName
import kotlinx.serialization.Serializable

@Serializable
data class SearchResponse(
```

```
8       val results: List<UltramanResponse>
9   )
```

4. Model/UltramanResponse.kt

Tabel 18. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.model
2
3   import kotlinx.serialization.SerialName
4   import kotlinx.serialization.Serializable
5
6   @Serializable
7   data class UltramanResponse(
8       val id: Int,
9       @SerialName("name") val name: String? = null,
10      @SerialName("title") val title: String? = null,
11      val overview: String,
12      @SerialName("poster_path") val posterPath: String?,
13
14      @SerialName("release_date") val releaseDate: String? = null,
15      @SerialName("first_air_date") val firstAirDate: String? =
    null
16  )
```

5. Model/UltramanMapper.kt

Tabel 19. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.model
2
3   fun UltramanResponse.toUltramanItem(): UltramanItem {
4       return UltramanItem(
5           id = id,
6           title = title ?: name ?: "Unknown Title",
7           description = overview,
```

```
        posterPath = posterPath,
        releaseDate = releaseDate,
        firstAirDate = firstAirDate
    )
}
```

6. Network/ TmdbApiService.kt

Tabel 20. Source Code Jawaban Soal 1 Modul 5

```
package com.example.ultraman.network

import com.example.ultraman.model.SearchResponse
import com.example.ultraman.model.UltramanResponse
import retrofit2.http.GET
import retrofit2.http.Path
import retrofit2.http.Query

interface TmdbApiService {

    @GET("tv/{id}")
    suspend fun getTvDetails(
        @Path("id") id: Int,
        @Query("api_key") apiKey: String
    ): UltramanResponse

    @GET("movie/{id}")
    suspend fun getMovieDetails(
        @Path("id") id: Int,
        @Query("api_key") apiKey: String
    ): UltramanResponse

    @GET("search/multi")
    suspend fun searchUltraman(
        @Query("query") query: String = "Ultraman",
```

```
26          @Query("api_key") apiKey: String,
27          @Query("page") page: Int = 1
28      ): SearchResponse
29
30  }
```

7. Network/ RetrofitInstance.kt

Tabel 21. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.network
2
3   import
    com.jakewharton.retrofit2.converter.kotlinx.serialization.
    asConverterFactory
4   import kotlinx.serialization.json.Json
5   import okhttp3.MediaType.Companion.toMediaType
6   import okhttp3.OkHttpClient
7   import retrofit2.Retrofit
8
9   object RetrofitInstance {
10      private const val BASE_URL =
    "https://api.themoviedb.org/3/"
11
12      private val json = Json {
13          ignoreUnknownKeys = true
14      }
15
16      private val client = OkHttpClient.Builder().build()
17
18      val api: TmdbApiService by lazy {
19          Retrofit.Builder()
20              .baseUrl(BASE_URL)
21              .addConverterFactory(json.asConverterFactory(
    "application/json".toMediaType()))
```

```
            .client(client)
            .build()
            .create(TmdbApiService::class.java)
    }
}
```

8. Data/ApiResponse.kt

Tabel 22. Source Code Jawaban Soal 1 Modul 5

```
package com.example.ultraman.data

sealed class ApiResponse<out T> {
    data class Success<T>(val data: T) : ApiResponse<T>()
    data class Error(val message: String) : ApiResponse<Nothing>()
    object Loading : ApiResponse<Nothing>()
}
```

9. Data/ UltramanDao.kt

Tabel 23. Source Code Jawaban Soal 1 Modul 5

```
package com.example.ultraman.data

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.example.ultraman.model.UltramanItem
import kotlinx.coroutines.flow.Flow

@Dao
interface UltramanDao {
    @Query("SELECT * FROM ultraman")
    fun getAll(): Flow<List<UltramanItem>>
```

```
14
15      @Insert(onConflict = OnConflictStrategy.REPLACE)
16      suspend fun insertAll(items: List<UltramanItem>)
17
18      @Query("UPDATE ultraman SET isFavorite = :isFav WHERE id =
    :id")
19      suspend fun setFavorite(id: Int, isFav: Boolean)
20
21      @Query("SELECT * FROM ultraman WHERE isFavorite = 1")
22      fun getFavorites(): Flow<List<UltramanItem>>
23
24      @Query("SELECT * FROM ultraman")
25      suspend fun getAllOnce(): List<UltramanItem>
26  }
```

10. Data/UltramanDatabase.kt

Tabel 24. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.data
2
3   import android.content.Context
4   import androidx.room.Database
5   import androidx.room.Room
6   import androidx.room.RoomDatabase
7   import com.example.ultraman.model.UltramanItem
8
9   @Database(entities = [UltramanItem::class], version = 3,
    exportSchema = false)
10  abstract class UltramanDatabase : RoomDatabase() {
11      abstract fun ultramanDao(): UltramanDao
12
13      companion object {
14          @Volatile
15          private var INSTANCE: UltramanDatabase? = null
```

```
16
17          fun getDatabase(context: Context): UltramanDatabase {
18              return INSTANCE ?: synchronized(this) {
19                  val instance = Room.databaseBuilder(
20                      context.applicationContext,
21                      UltramanDatabase::class.java,
22                      "ultraman_db"
23                  ).fallbackToDestructiveMigration()
24                  .build()
25                  INSTANCE = instance
26                  instance
27              }
28          }
29      }
30  }
```

11. Repository/UltramanRepository.kt

Tabel 25. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.repository
2
3   import android.util.Log
4   import com.example.ultraman.data.ApiResponse
5   import com.example.ultraman.data.UltramanDao
6   import com.example.ultraman.model.UltramanItem
7   import com.example.ultraman.model.toUltramanItem
8   import com.example.ultraman.network.RetrofitInstance
9   import kotlinx.coroutines.flow.Flow
10  import kotlinx.coroutines.flow.flow
11
12  class UltramanRepository(private val dao: UltramanDao) {
13
14      val ultramanItems: Flow<List<UltramanItem>> = dao.getAll()
15      val      favoriteItems:      Flow<List<UltramanItem>>      =
    dao.getFavorites()
```

```
16
17      fun refreshData(apiKey: String): Flow<ApiResponse<List<UltramanItem>>> = flow {
18          emit(ApiResponse.Loading)
19
20          try {
21              val searchResponse = RetrofitInstance.api.searchUltraman(apiKey = apiKey)
22              val newItems = searchResponse.results.mapNotNull { it.toUltramanItem() }
23
24              val oldItems = dao.getAllOnce()
25              val mergedItems = newItems.map { newItem ->
26                  val old = oldItems.find { it.id == newItem.id }
27                  newItem.copy(isFavorite = old?.isFavorite ?: false)
28              }
29
30              dao.insertAll(mergedItems.take(20))
31              emit(ApiResponse.Success(mergedItems.take(20)))
32
33          } catch (e: Exception) {
34              Log.e("UltramanRepository", "Error: ${e.message}")
35              emit(ApiResponse.Error(e.message ?: "Terjadi kesalahan"))
36          }
37      }
38
39      suspend fun toggleFavorite(id: Int, currentValue: Boolean) {
40          dao.setFavorite(id, !currentValue)
41      }
42
43      fun getFavorites(): Flow<List<UltramanItem>> {
```

```
44          return dao.getFavorites()
45      }
46  }
```

12. ui/ViewModel/UltramanViewModel.kt

Tabel 26. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.ui.ViewModel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import com.example.ultraman.data.ApiResponse
6   import com.example.ultraman.model.UltramanItem
7   import com.example.ultraman.repository.UltramanRepository
8   import kotlinx.coroutines.flow.MutableStateFlow
9   import kotlinx.coroutines.flow.StateFlow
10  import kotlinx.coroutines.flow.collectLatest
11  import kotlinx.coroutines.launch
12
13  class UltramanViewModel(private val repository:
    UltramanRepository) : ViewModel() {
14
15      private val _ultramanList =
    MutableStateFlow<List<UltramanItem>>(emptyList())
16      val ultramanList: StateFlow<List<UltramanItem>> =
    _ultramanList
17
18      private val _favoriteList =
    MutableStateFlow<List<UltramanItem>>(emptyList())
19      val favoriteList: StateFlow<List<UltramanItem>> =
    _favoriteList
20
21      private val _uiState = MutableStateFlow<ApiResponse
    <List<UltramanItem>>>(ApiResponse.Loading)
```

```
22      val uiState: StateFlow<ApiResponse<List
    <UltramanItem>>> = _uiState
23
24      private val _favoriteState =
    MutableStateFlow<ApiResponse<List<UltramanItem>>>
    (ApiResponse.Loading)
25      val favoriteState:
    StateFlow<ApiResponse<List<UltramanItem>>> = _favoriteState
26
27      init {
28          refreshData()
29          observeLocalData()
30      }
31
32      private fun observeLocalData() {
33          viewModelScope.launch {
34              repository.ultramanItems.collectLatest {
35                  _ultramanList.value = it
36                  _uiState.value = ApiResponse.Success(it)
37              }
38          }
39
40          viewModelScope.launch {
41              try {
42                  repository.favoriteItems.collectLatest {
43                      _favoriteList.value = it
44                      _favoriteState.value =
    ApiResponse.Success(it)
45                  }
46              } catch (e: Exception) {
47                  _favoriteState.value =
    ApiResponse.Error(e.message ?: "Unknown error")
48              }
49          }
```

```
50      }
51
52      fun refreshData(apiKey: String =
    "5747e2247c220f5aac1c7654bfa77d4b") {
53          viewModelScope.launch {
54              repository.refreshData(apiKey).collectLatest {
55                  _uiState.value = it
56              }
57          }
58      }
59
60      fun getItemById(id: Int): UltramanItem? {
61          return _ultramanList.value.find { it.id == id }
62      }
63
64      fun toggleFavorite(item: UltramanItem) {
65          viewModelScope.launch {
66              repository.toggleFavorite(item.id, item.isFavorite)
67          }
68      }
69  }
```

13. ui/ViewModel/UltramanViewModelFactory.kt

Tabel 27. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.ui.ViewModel
2
3   import android.content.Context
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6   import com.example.ultraman.data.UltramanDatabase
7   import com.example.ultraman.repository.UltramanRepository
8
```

```
9  class UltramanViewModelFactory(private val context: Context) :
   ViewModelProvider.Factory {
10     override fun <T : ViewModel> create(modelClass: Class<T>):
   T {
11         val                          dao                          =
   UltramanDatabase.getDatabase(context).ultramanDao()
12         val repository = UltramanRepository(dao)
13         @Suppress("UNCHECKED_CAST")
14         return UltramanViewModel(repository) as T
15     }
16 }
```

14. ui/screens/ListScreen.kt

Tabel 28. Source Code Jawaban Soal 1 Modul 5

```
1  package com.example.ultraman.ui.screens
2
3  import android.content.Intent
4  import androidx.compose.foundation.background
5  import androidx.compose.foundation.layout.*
6  import androidx.compose.foundation.lazy.LazyColumn
7  import androidx.compose.foundation.lazy.items
8  import androidx.compose.foundation.shape.CircleShape
9  import androidx.compose.foundation.shape.RoundedCornerShape
10 import androidx.compose.material.icons.Icons
11 import androidx.compose.material.icons.filled.Favorite
12 import androidx.compose.material.icons.filled.FavoriteBorder
13 import androidx.compose.material3.*
14 import androidx.compose.runtime.Composable
15 import androidx.compose.runtime.collectAsState
16 import androidx.compose.ui.Alignment
17 import androidx.compose.ui.Modifier
18 import androidx.compose.ui.draw.clip
19 import androidx.compose.ui.graphics.Color
```

```
20  import androidx.compose.ui.layout.ContentScale
21  import androidx.compose.ui.text.SpanStyle
22  import androidx.compose.ui.text.buildAnnotatedString
23  import androidx.compose.ui.text.font.FontWeight
24  import androidx.compose.ui.text.style.TextOverflow
25  import androidx.compose.ui.text.withStyle
26  import androidx.compose.ui.unit.dp
27  import androidx.core.net.toUri
28  import androidx.navigation.NavHostController
29  import com.example.ultraman.ui.ViewModel.UltramanViewModel
30  import com.example.ultraman.data.ApiResponse
31
32
33  @Composable
34  fun ListScreen(
35      navController: NavHostController,
36      viewModel: UltramanViewModel,
37      modifier: Modifier = Modifier
38  ) {
39      val uiState = viewModel.uiState.collectAsState().value
40
41      when (uiState) {
42          is ApiResponse.Loading -> {
43              Box(
44                  modifier = Modifier.fillMaxSize(),
45                  contentAlignment = Alignment.Center
46              ) {
47                  CircularProgressIndicator()
48              }
49          }
50
51          is ApiResponse.Error -> {
52              Box(
```

```
53                  modifier                                        =
    Modifier.fillMaxSize().padding(16.dp),
54                  contentAlignment = Alignment.Center
55              ) {
56                  Text("Terjadi  kesalahan:  ${uiState.message}",
    color = Color.Red)
57              }
58          }
59
60          is ApiResponse.Success -> {
61              val ultramanList = uiState.data
62              LazyColumn(
63                  modifier = modifier
64                      .fillMaxSize()
65                      .padding(8.dp)
66              ) {
67                  items(ultramanList) { item ->
68                      Card(
69                          shape = RoundedCornerShape(16.dp),
70                          elevation                                   =
    CardDefaults.cardElevation(8.dp),
71                          modifier = Modifier
72                              .fillMaxWidth()
73                              .padding(vertical = 8.dp)
74                      ) {
75                          Row(
76                              modifier = Modifier
77                                  .padding(16.dp)
78                                  .fillMaxWidth()
79                          ) {
80                              Box(
81                                  modifier = Modifier
82                                      .height(150.dp)
83                                      .width(100.dp)
```

```
84                              ) {
85                                  GlideImage(
86                                      imageUrl                =
    item.getPosterUrl().toString(),
87                                      contentDescription      =
    item.title,
88                                      modifier = Modifier
89                                          .matchParentSize()
90
    .clip(RoundedCornerShape(12.dp)),
91                                      contentScale            =
    ContentScale.Crop
92                                  )
93
94                                  IconButton(
95                                      onClick        =        {
    viewModel.toggleFavorite(item) },
96                                      modifier = Modifier
97
    .align(Alignment.TopStart)
98                                          .padding(4.dp)
99
    .background(Color.White.copy(alpha    =    0.5f),    shape    =
    CircleShape)
100                                 ) {
101                                     Icon(
102                                         imageVector    =    if
    (item.isFavorite)          Icons.Default.Favorite          else
    Icons.Default.FavoriteBorder,
103                                         contentDescription   =
    "Favorite",
104                                         tint = Color.Red
105                                     )
106                                 }
```

```
107                             }
108
109                             Spacer(modifier                 =
    Modifier.width(16.dp))
110
111                             Column(
112                                 modifier = Modifier.weight(1f)
113                             ) {
114                                 Row(
115                                     verticalAlignment       =
    Alignment.CenterVertically,
116                                     horizontalArrangement   =
    Arrangement.SpaceBetween,
117                                     modifier                =
    Modifier.fillMaxWidth()
118                                 ) {
119                                     Text(
120                                         text                =
    item.title.take(20) + if (item.title.length > 20) "…" else "",
121                                         style               =
    MaterialTheme.typography.titleMedium,
122                                         fontWeight          =
    FontWeight.Bold
123                                     )
124                                     Text(
125                                         text = item.year,
126                                         style               =
    MaterialTheme.typography.bodySmall,
127                                         color = Color.Gray
128                                     )
129                                 }
130
131                                 Spacer(modifier             =
    Modifier.height(6.dp))
```

```
132
133                                 Row(
134                                     modifier                =
    Modifier.fillMaxWidth()
135                                 ) {
136                                     Text(
137                                         text = "Info: ",
138                                         style                =
    MaterialTheme.typography.bodySmall,
139                                         fontWeight           =
    FontWeight.Bold
140                                     )
141                                     Text(
142                                         text                 =
    item.description,
143                                         style                =
    MaterialTheme.typography.bodySmall,
144                                         maxLines = 4,
145                                         overflow             =
    TextOverflow.Ellipsis
146                                     )
147                                 }
148
149                                 Spacer(modifier              =
    Modifier.height(12.dp))
150
151                                 Row(
152                                     horizontalArrangement    =
    Arrangement.spacedBy(30.dp)
153                                 ) {
154                                     Button(
155                                         onClick = {
156                                             val    intent    =
    Intent(Intent.ACTION_VIEW, item.getTmdbLink().toUri())
```

```
157 navController.context.startActivity(intent)
158                                         },
159                                         modifier              =
    Modifier.weight(1f)
160                                     ) {
161                                         Text("More info")
162                                     }
163
164                                     Button(
165                                         onClick = {
166
    navController.navigate("detail/${item.id}")
167                                         },
168                                         modifier              =
    Modifier.weight(1f)
169                                     ) {
170                                         Text("Detail")
171                                     }
172                                 }
173                             }
174                         }
175                     }
176                 }
177             }
178         }
179     }
180 }
```

15. ui/screens/DetailScreen.kt

Tabel 29. Source Code Jawaban Soal 1 Modul 5

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

```
3  import android.util.Log
4  import androidx.compose.foundation.layout.*
5  import androidx.compose.foundation.rememberScrollState
6  import androidx.compose.foundation.shape.RoundedCornerShape
7  import androidx.compose.foundation.verticalScroll
8  import androidx.compose.material3.MaterialTheme
9  import androidx.compose.material3.Text
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.draw.clip
13 import androidx.compose.ui.layout.ContentScale
14 import androidx.compose.ui.text.font.FontWeight
15 import androidx.compose.ui.text.style.TextAlign
16 import androidx.compose.ui.unit.dp
17 import com.example.ultraman.ui.ViewModel.UltramanViewModel
18
19 @Composable
20 fun DetailScreen(itemId: Int?, viewModel: UltramanViewModel) {
21     val item = itemId?.let { viewModel.getItemById(it) }
22     Log.d("DetailScreen", "Navigasi ke DetailScreen untuk ID:
   $itemId - ${item?.title}")
23
24     item?.let {
25         Column(
26             modifier = Modifier
27                 .fillMaxSize()
28                 .padding(16.dp)
29
   .padding(WindowInsets.systemBars.asPaddingValues())
30                 .verticalScroll(rememberScrollState())
31         ) {
32             it.getPosterUrl()?.let { posterUrl ->
33                 GlideImage(
34                     imageUrl = posterUrl,
```

```
35                      contentDescription = it.title,
36                      modifier = Modifier
37                          .fillMaxWidth()
38                          .height(600.dp)
39                          .clip(RoundedCornerShape(16.dp)),
40                      contentScale = ContentScale.Crop
41                  )
42              }
43
44              Spacer(modifier = Modifier.height(12.dp))
45
46              Text(
47                  text = it.title,
48                  style = MaterialTheme.typography.headlineSmall,
49                  fontWeight = FontWeight.Bold
50              )
51              Spacer(modifier = Modifier.height(8.dp))
52
53              Text( text = "Tahun: ",
54                    fontWeight = FontWeight.Bold
55              )
56              Text(it.year)
57              Text(
58                  text = "Sinopsis:",
59                  style = MaterialTheme.typography.titleSmall,
60                  fontWeight = FontWeight.Bold
61              )
62              Text(
63                  text = it.description,
64                  style = MaterialTheme.typography.bodyMedium,
65                  textAlign = TextAlign.Justify
66              )
67          }
68      } ?: run {
```

```
69          Column(
70              modifier = Modifier
71                  .fillMaxSize()
72                  .padding(16.dp),
73              verticalArrangement = Arrangement.Center
74          ) {
75              Text(
76                  text = "Data tidak ditemukan.",
77                  style = MaterialTheme.typography.bodyLarge,
78                  modifier = Modifier.fillMaxWidth(),
79                  textAlign = TextAlign.Center
80              )
81          }
82      }
83  }
```

16. ui/screens/favoritescreen.kt

Tabel 30. Source Code Jawaban Soal 1 Modul 5

```
1   package com.example.ultraman.ui.screens
2
3   import android.content.Intent
4   import androidx.compose.foundation.background
5   import androidx.compose.foundation.layout.*
6   import androidx.compose.foundation.lazy.LazyColumn
7   import androidx.compose.foundation.lazy.items
8   import androidx.compose.foundation.shape.CircleShape
9   import androidx.compose.foundation.shape.RoundedCornerShape
10  import androidx.compose.material.icons.Icons
11  import androidx.compose.material.icons.filled.Favorite
12  import androidx.compose.material.icons.filled.FavoriteBorder
13  import androidx.compose.material3.*
14  import androidx.compose.runtime.Composable
15  import androidx.compose.runtime.collectAsState
```

```
16  import androidx.compose.ui.Alignment
17  import androidx.compose.ui.Modifier
18  import androidx.compose.ui.draw.clip
19  import androidx.compose.ui.graphics.Color
20  import androidx.compose.ui.layout.ContentScale
21  import androidx.compose.ui.text.font.FontWeight
22  import androidx.compose.ui.text.style.TextOverflow
23  import androidx.compose.ui.unit.dp
24  import androidx.core.net.toUri
25  import androidx.navigation.NavHostController
26  import com.example.ultraman.ui.ViewModel.UltramanViewModel
27  import com.example.ultraman.data.ApiResponse
28  import com.example.ultraman.ui.screens.GlideImage
29
30  @Composable
31  fun FavoriteScreen(
32      navController: NavHostController,
33      viewModel: UltramanViewModel,
34      modifier: Modifier = Modifier
35  ) {
36      val                          uiState                          =
    viewModel.favoriteState.collectAsState().value
37
38      when (uiState) {
39          is ApiResponse.Loading -> {
40              Box(
41                  modifier = Modifier.fillMaxSize(),
42                  contentAlignment = Alignment.Center
43              ) {
44                  CircularProgressIndicator()
45              }
46          }
47
48          is ApiResponse.Error -> {
```

```
49              Box(
50                  modifier                                        =
    Modifier.fillMaxSize().padding(16.dp),
51                  contentAlignment = Alignment.Center
52              ) {
53                  Text("Terjadi  kesalahan:  ${uiState.message}",
    color = Color.Red)
54              }
55          }
56
57          is ApiResponse.Success -> {
58              val favoriteList = uiState.data
59
60              if (favoriteList.isEmpty()) {
61                  Box(
62                      modifier = Modifier.fillMaxSize(),
63                      contentAlignment = Alignment.Center
64                  ) {
65                      Text("Belum ada yang difavoritkan", color
    = Color.Gray)
66                  }
67              } else {
68                  LazyColumn(
69                      modifier = modifier
70                          .fillMaxSize()
71                          .padding(8.dp)
72                  ) {
73                      items(favoriteList) { item ->
74                          Card(
75                              shape = RoundedCornerShape(16.dp),
76                              elevation                           =
    CardDefaults.cardElevation(8.dp),
77                              modifier = Modifier
78                                  .fillMaxWidth()
```

```
79                                      .padding(vertical = 8.dp)
80                              ) {
81                                  Row(
82                                      modifier = Modifier
83                                          .padding(16.dp)
84                                          .fillMaxWidth()
85                                  ) {
86                                      Box(
87                                          modifier = Modifier
88                                              .height(150.dp)
89                                              .width(100.dp)
90                                      ) {
91                                          GlideImage(
92                                              imageUrl              =
    item.getPosterUrl().toString(),
93                                              contentDescription    =
    item.title,
94                                              modifier = Modifier
95                                                  .matchParentSize()
96
    .clip(RoundedCornerShape(12.dp)),
97                                              contentScale          =
    ContentScale.Crop
98                                          )
99
100                                         IconButton(
101                                             onClick       =       {
    viewModel.toggleFavorite(item) },
102                                             modifier = Modifier

103 .align(Alignment.TopStart)
104                                                 .padding(4.dp)
```

```
105 .background(Color.White.copy(alpha = 0.5f), shape =
    CircleShape)
106                                     ) {
107                                         Icon(
108                                             imageVector = if
    (item.isFavorite)          Icons.Default.Favorite          else
    Icons.Default.FavoriteBorder,
109                                             contentDescription
    = "Favorite",
110                                             tint = Color.Red
111                                         )
112                                     }
113                                 }
114
115                                 Spacer(modifier              =
    Modifier.width(16.dp))
116
117                                 Column(
118                                     modifier                 =
    Modifier.weight(1f)
119                                 ) {
120                                     Row(
121                                         verticalAlignment    =
    Alignment.CenterVertically,
122                                         horizontalArrangement
    = Arrangement.SpaceBetween,
123                                         modifier             =
    Modifier.fillMaxWidth()
124                                     ) {
125                                         Text(
126                                             text             =
    item.title.take(20) + if (item.title.length > 20) "…" else "",
```

```
127                                         style             =
    MaterialTheme.typography.titleMedium,
128                                         fontWeight        =
    FontWeight.Bold
129                                     )
130                                     Text(
131                                         text = item.year,
132                                         style             =
    MaterialTheme.typography.bodySmall,
133                                         color = Color.Gray
134                                     )
135                                 }
136
137                                 Spacer(modifier           =
    Modifier.height(6.dp))
138
139                                 Row(
140                                     modifier              =
    Modifier.fillMaxWidth()
141                                 ) {
142                                     Text(
143                                         text = "Info: ",
144                                         style             =
    MaterialTheme.typography.bodySmall,
145                                         fontWeight        =
    FontWeight.Bold
146                                     )
147                                     Text(
148                                         text              =
    item.description,
149                                         style             =
    MaterialTheme.typography.bodySmall,
150                                         maxLines = 4,
```

```
151                                             overflow          =
    TextOverflow.Ellipsis
152                                         )
153                                     }
154
155                                     Spacer(modifier           =
    Modifier.height(12.dp))
156
157                                     Row(
158                                         horizontalArrangement
    = Arrangement.spacedBy(30.dp)
159                                     ) {
160                                         Button(
161                                             onClick = {
162                                                 val  intent  =
    Intent(Intent.ACTION_VIEW, item.getTmdbLink().toUri())
163
    navController.context.startActivity(intent)
164                                             },
165                                             modifier          =
    Modifier.weight(1f)
166                                         ) {
167                                             Text("More info")
168                                         }
169
170                                         Button(
171                                             onClick = {
172
    navController.navigate("detail/${item.id}")
173                                             },
174                                             modifier          =
    Modifier.weight(1f)
175                                         ) {
176                                             Text("Detail")
```

```
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
```

17. ui/screens/GlideImage.kt

Tabel 31. Source Code Jawaban Soal 1 Modul 5

```
package com.example.ultraman.ui.screens

import android.widget.ImageView
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.viewinterop.AndroidView
import com.bumptech.glide.Glide
import androidx.compose.ui.layout.ContentScale

@Composable
fun GlideImage(
    imageUrl: String,
    contentDescription: String?,
    modifier: Modifier = Modifier,
    contentScale: ContentScale = ContentScale.Crop
) {
    val context = LocalContext.current
    AndroidView(
```

```
20          factory = {
21              ImageView(context).apply {
22                  scaleType = when (contentScale) {
23                      ContentScale.Crop                          ->
    ImageView.ScaleType.CENTER_CROP
24                      ContentScale.Fit                           ->
    ImageView.ScaleType.FIT_CENTER
25                      else -> ImageView.ScaleType.CENTER_CROP
26                  }
27                  contentDescription?.let                         {
    this.contentDescription = it }
28              }
29          },
30          update = {
31              Glide.with(context)
32                  .load(imageUrl)
33                  .into(it)
34          },
35          modifier = modifier
36      )
37  }
```

18. ui/components/BottomBarNavigation.kt

Tabel 32. Source Code Jawaban Soal 1 Modul 5

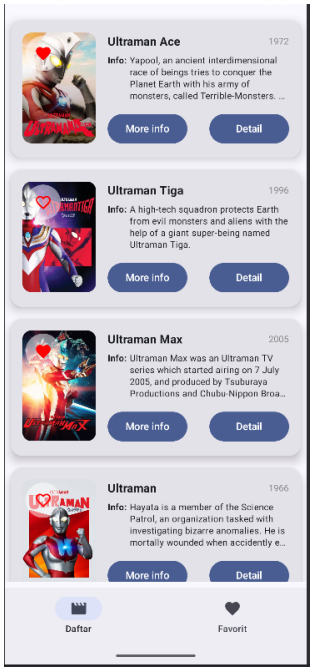
```
1   package com.example.ultraman.ui.components
2
3   import androidx.compose.material.icons.Icons
4   import androidx.compose.material.icons.filled.Favorite
5   import androidx.compose.material.icons.filled.Movie
6   import androidx.compose.material3.*
7   import androidx.compose.runtime.Composable
8   import androidx.compose.ui.graphics.vector.ImageVector
9   import androidx.navigation.NavController
```

```
10 import androidx.navigation.compose.currentBackStackEntryAsState
11
12 data class BottomNavItem(
13     val title: String,
14     val icon: ImageVector,
15     val route: String
16 )
17
18 val bottomNavItems = listOf(
19     BottomNavItem("Daftar", Icons.Default.Movie, "list"),
20     BottomNavItem("Favorit",            Icons.Default.Favorite,
   "favorite")
21 )
22
23 @Composable
24 fun BottomBarNavigation(navController: NavController) {
25     NavigationBar {
26         val                 navBackStackEntry                =
   navController.currentBackStackEntryAsState().value
27         val                   currentRoute                   =
   navBackStackEntry?.destination?.route
28
29         bottomNavItems.forEach { item ->
30             NavigationBarItem(
31                 selected = currentRoute == item.route,
32                 onClick = {
33                     if (currentRoute != item.route) {
34                         navController.navigate(item.route) {
35                             popUpTo("list") { inclusive = false
   }
36                             launchSingleTop = true
37                         }
38                     }
39                 },
```

```
40                  icon = { Icon(imageVector = item.icon,
    contentDescription = item.title) },
41                  label = { Text(item.title) }
42              )
43          }
44      }
45  }
```

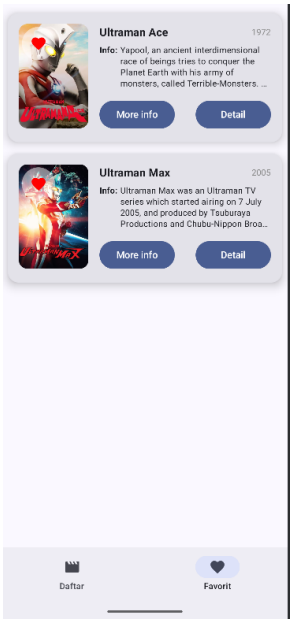
B. Output Program

- Tampilan halaman daftar film ultraman :



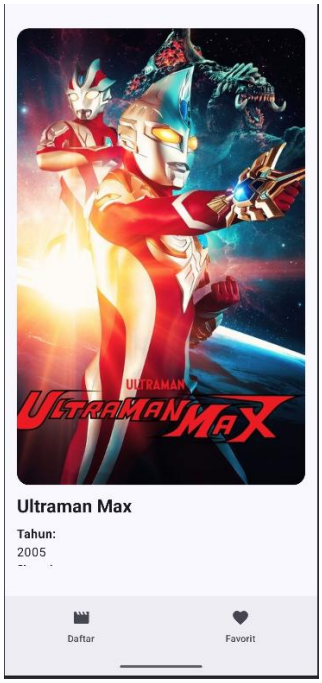
Gambar 29. Screenshot Hasil Jawaban Soal 1 Modul 5

- Tampilan Halaman film favorit (menampilkan film ultraman yang ditandai favorit) :



Gambar 30. Screenshot Hasil Jawaban Soal 1 Modul 5

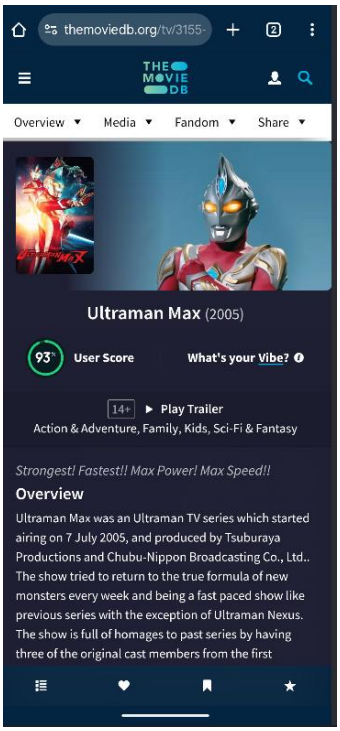
- Tampilan Halaman Detail :



Gambar 31. Screenshot Hasil Jawaban Soal 1 Modul 5

Gambar 32. Screenshot Hasil Jawaban Soal 1 Modul 5

- Ketika Klik More Info :



Gambar 33. Screenshot Hasil Jawaban Soal 1 Modul 5

C. Pembahasan

1. MainActivity.kt

Di dalam MainActivity.kt, saya membuat struktur utama aplikasi Android bertema Ultraman dengan menggunakan Jetpack Compose dan sistem navigasi berbasis NavController. Pertama, saya extend ComponentActivity, lalu di dalam onCreate, saya memanggil setContent untuk menyusun UI utama aplikasi dengan menggunakan UltramanTheme yang telah saya definisikan sebelumnya.

Untuk kebutuhan navigasi, saya menggunakan rememberNavController() dan saya siapkan UltramanViewModel lewat viewModel() dengan custom factory UltramanViewModelFactory agar bisa mengakses context dari aplikasi (diperlukan untuk Room database). Selanjutnya, saya panggil fungsi UltramanApp() dan mengoperkan navController dan viewModel ke dalamnya.

Di dalam fungsi UltramanApp, saya menggunakan Scaffold untuk menyusun layout dasar Compose dengan BottomBarNavigation sebagai komponen bottomBar-nya. Dengan begitu, pengguna bisa berpindah antar halaman utama aplikasi menggunakan bottom navigation.

Kemudian, saya atur navigasi menggunakan NavHost. Start destination-nya saya tetapkan ke "list" agar aplikasi menampilkan daftar Ultraman saat pertama kali dibuka. Saya definisikan tiga rute utama:

- "list" Menampilkan ListScreen, yaitu daftar semua serial dan film Ultraman.
- "favorite" Menampilkan FavoriteScreen, yang berisi daftar tontonan yang ditandai sebagai favorit oleh pengguna.
- "detail/{itemId}" Menampilkan DetailScreen berdasarkan ID item yang diklik. Rute ini menggunakan parameter itemId bertipe Int, dan saya ambil nilainya dari backStackEntry.arguments.

2. UltramanItem.kt

Pada file UltramanItem.kt, saya mendefinisikan sebuah data class bernama UltramanItem yang merepresentasikan satu entitas Ultraman, baik itu film maupun serial TV, dan ini digunakan di seluruh aplikasi saya. Kelas ini saya tandai dengan anotasi @Entity dari Room

karena datanya juga saya simpan secara lokal ke database, dan saya beri nama tabelnya "ultraman". Saya menggunakan @PrimaryKey untuk menetapkan id sebagai kunci utama, sesuai dengan ID dari TMDB.

Karena saya juga mengambil data dari TMDB API, saya menggunakan anotasi @Serializable dari Kotlinx Serialization, dan saya map field JSON seperti "overview", "poster_path", "release_date", dan "first_air_date" ke properti Kotlin menggunakan @SerialName, agar bisa langsung diparsing otomatis.

Properti isFavorite saya tambahkan sebagai penanda apakah item ini sudah ditandai favorit oleh user atau belum, dengan default false. Lalu, saya buat properti turunan year untuk mengambil tahun rilis, baik dari releaseDate (jika itu film) atau firstAirDate (jika itu serial), dan kalau dua-duanya kosong, akan tampil "????".

Selain itu, saya juga buat dua fungsi utilitas: getPosterUrl() yang menghasilkan URL lengkap ke gambar poster dari TMDB, dan getTmdbLink() yang menentukan link ke halaman TMDB dari item tersebut, membedakan antara movie dan TV berdasarkan apakah releaseDate ada atau tidak.

3. SearchResponse.kt

Pada file SearchResponse.kt, saya membuat sebuah data class bernama SearchResponse yang digunakan untuk memetakan respons JSON dari API pencarian TMDB. Saya beri anotasi @Serializable agar bisa langsung diparsing menggunakan Kotlinx Serialization.

Respons dari TMDB untuk endpoint pencarian biasanya berupa objek JSON yang memiliki properti results, yang isinya adalah daftar hasil pencarian dalam bentuk array. Di sini saya map properti results tersebut ke properti results di Kotlin, yang bertipe List<UltramanResponse>, karena masing-masing item hasil pencarian nantinya akan direpresentasikan oleh kelas UltramanResponse.kt.

Dengan begini, saat saya melakukan request pencarian, saya bisa langsung parsing respons JSON-nya menjadi objek SearchResponse, dan mengambil daftar hasil pencariannya lewat properti results.

4. UltramanResponse.kt

Pada file UltramanResponse.kt, saya membuat data class UltramanResponse sebagai representasi dari satu item hasil respons API TMDB, baik itu berupa film maupun serial. Saya beri anotasi @Serializable agar data ini bisa langsung diparsing dari JSON menggunakan Kotlinx Serialization.

Properti id adalah ID unik dari TMDB yang selalu tersedia. Karena TMDB membedakan antara film (title) dan serial TV (name), saya siapkan kedua field tersebut sebagai properti opsional (String?) dan memetakannya ke JSON dengan @SerialName. Dengan begitu, saya bisa menangani kedua jenis konten tanpa harus membuat dua data class terpisah.

Field overview adalah deskripsi dari film atau serial, sedangkan poster_path saya map ke posterPath yang berisi path gambar poster. Lalu saya juga menambahkan dua tanggal berbeda: release_date (untuk film) dan first_air_date (untuk serial), yang masing-masing saya map ke releaseDate dan firstAirDate.

Data class ini saya gunakan sebagai struktur mentah yang langsung dihasilkan dari API sebelum diubah ke model utama aplikasi saya, yaitu UltramanItem. Biasanya proses konversinya dilakukan melalui mapper.

5. UltramanMapper.kt

Pada file ini, saya membuat sebuah ekstensi fungsi bernama toUltramanItem() bertugas mengonversi objek UltramanResponse yang merupakan hasil mentah dari API TMDB menjadi UltramanItem, yaitu model utama yang saya gunakan di seluruh aplikasi, termasuk untuk penyimpanan di Room dan penampilan di UI.

Dalam fungsi ini, saya mengambil properti id, overview, posterPath, releaseDate, dan firstAirDate langsung dari UltramanResponse dan meneruskannya apa adanya ke dalam UltramanItem. Untuk bagian title, karena TMDB kadang menggunakan title (untuk film) dan kadang name (untuk serial), saya prioritaskan nilai dari title, lalu fallback ke name, dan jika keduanya tidak tersedia, saya isi dengan string default "Unknown Title".

Fungsi ini saya buat agar proses konversi data menjadi lebih bersih, modular, dan bisa digunakan berulang kali setiap kali saya menerima data dari API. Dengan begitu, saya bisa

menjaga agar UI dan database hanya berinteraksi dengan satu model data yang konsisten, yaitu UltramanItem.

6. TmdbApiService.kt

Di file TmdbApiService.kt, saya mendefinisikan interface TmdbApiService sebagai kontrak komunikasi antara aplikasi saya dan TMDB API menggunakan Retrofit. Interface ini berisi deklarasi beberapa endpoint yang saya butuhkan untuk mengambil data terkait Ultraman, baik itu berupa film, serial, maupun hasil pencarian gabungan.

Fungsi getTvDetails() dan getMovieDetails() masing-masing digunakan untuk mengambil detail lengkap dari sebuah TV show atau film berdasarkan ID-nya. Saya gunakan anotasi @GET("tv/{id}") dan @GET("movie/{id}") dengan parameter dinamis {id} yang diisi lewat anotasi @Path. Selain itu, karena TMDB memerlukan API key, saya sertakan parameter @Query("api_key") agar bisa dikirimkan sebagai query parameter dalam setiap request. Respons dari keduanya akan langsung dipetakan ke objek UltramanResponse.

Sementara itu, fungsi searchUltraman() saya buat untuk melakukan pencarian dengan query default "Ultraman", namun tetap fleksibel karena bisa diganti dengan kata kunci lain. Saya pakai endpoint search/multi karena endpoint ini bisa mencari baik film maupun TV sekaligus. Parameter page saya beri default 1 untuk mendukung pagination nanti kalau dibutuhkan. Respons dari fungsi ini saya petakan ke SearchResponse, yang berisi daftar UltramanResponse.

Interface ini menjadi pondasi utama komunikasi jaringan di aplikasi saya, dan akan diimplementasikan lewat instance Retrofit untuk mengambil data dari TMDB.

7. RetrofitInstance.kt

Di file RetrofitInstance.kt, saya membuat sebuah singleton object bernama RetrofitInstance yang berfungsi untuk menyediakan instance Retrofit yang sudah dikonfigurasi dengan baik untuk berkomunikasi dengan TMDB API. Tujuannya supaya saya bisa menggunakan satu instance Retrofit secara global di seluruh aplikasi tanpa perlu membuat ulang setiap kali.

Pertama, saya tentukan BASE_URL ke "https://api.themoviedb.org/3/", yaitu base URL resmi dari TMDB API versi 3. Kemudian, saya buat konfigurasi Json dari Kotlinx Serialization dengan opsi ignoreUnknownKeys = true. Pengaturan ini penting agar aplikasi

tidak error saat TMDB mengirim field yang tidak saya definisikan dalam model Kotlin — ini membuat parsing jadi lebih fleksibel.

Untuk client-nya, saya gunakan OkHttpClient default tanpa konfigurasi tambahan. Tapi kalau nanti perlu logging atau interceptor untuk debugging, saya bisa tambahkan di sini.

Lalu, saya buat instance Retrofit menggunakan Retrofit.Builder(), di mana saya pasang baseUrl, tambahkan converter dari Kotlinx Serialization (asConverterFactory()), dan set client sebagai HTTP client-nya. Setelah itu, saya panggil .create(TmdbApiService::class.java) untuk menghasilkan implementasi dari interface API yang sudah saya buat sebelumnya.

Akhirnya, properti api dideklarasikan menggunakan by lazy, jadi instance Retrofit hanya dibuat saat pertama kali digunakan. Dengan pendekatan ini, saya bisa dengan mudah mengakses RetrofitInstance.api di mana pun untuk memanggil endpoint TMDB secara efisien dan terpusat.

8. ApiResponse.kt

Pada file ApiResponse.kt, saya membuat sebuah kelas sealed bernama ApiResponse yang digunakan sebagai pembungkus hasil dari operasi jaringan (network request) di aplikasi saya. Tujuan saya membuat ini adalah untuk menangani status data secara reaktif baik saat data sedang dimuat (Loading), berhasil diambil (Success), atau gagal (Error) dalam satu struktur yang konsisten.

Karena ini adalah sealed class, saya bisa memastikan bahwa hanya ada tiga kemungkinan bentuk respons:

- Success<T> berisi data yang berhasil diambil, dibungkus dalam properti data. Ini bertipe generic agar bisa digunakan untuk berbagai jenis data, misalnya List<UltramanItem> atau objek tunggal seperti UltramanItem.
- Error digunakan ketika terjadi kegagalan saat mengambil data dari API. Saya hanya butuh pesan error-nya, jadi saya simpan dalam properti message.
- Loading merepresentasikan status saat request masih berjalan — biasanya saya pakai ini untuk menampilkan indikator loading di UI.

Dengan pola ini, saya bisa mengelola UI berdasarkan state secara lebih bersih di ViewModel dan Composable, misalnya menampilkan progress bar saat ApiResponse.Loading, menampilkan daftar saat Success, dan pesan kesalahan saat Error. Pendekatan ini juga mempermudah penggunaan Flow atau LiveData karena hanya perlu mengobservasi satu objek respons yang mencakup semua kemungkinan.

9. UltramanDao.kt

Pada file UltramanDao.kt, saya mendefinisikan interface UltramanDao yang menjadi data access object (DAO) untuk Room Database. Interface ini berisi fungsi-fungsi yang saya gunakan untuk mengelola data UltramanItem yang disimpan secara lokal di SQLite melalui Room. Dengan kata lain, inilah jembatan antara database dan layer ViewModel/repository.

Pertama, getAll() saya gunakan untuk mengambil semua data Ultraman dari tabel ultraman dan membungkus hasilnya dalam Flow<List<UltramanItem>>, sehingga saya bisa mengobservasi data ini secara reaktif di UI, cocok untuk penggunaan Jetpack Compose.

Fungsi insertAll() dipakai untuk menyimpan daftar Ultraman ke database. Saya beri strategi OnConflictStrategy.REPLACE, artinya kalau ada data dengan ID yang sama, maka akan ditimpa. Ini penting untuk sinkronisasi dengan API, supaya data lama yang sudah kadaluarsa bisa diperbarui.

Kemudian setFavorite() adalah fungsi yang saya gunakan untuk mengubah status favorit dari sebuah item berdasarkan ID-nya. Fungsi ini penting untuk fitur “tandai sebagai favorit” yang ada di aplikasi.

Selanjutnya, getFavorites() mengambil hanya data yang isFavorite = 1, dan membungkus hasilnya dalam Flow juga, agar UI bisa menampilkan daftar favorit secara real-time.

Terakhir, getAllOnce() adalah versi non-reaktif dari getAll(), hasilnya langsung berupa List<UltramanItem> biasa. Ini saya gunakan ketika hanya butuh snapshot satu kali, misalnya saat sinkronisasi awal, tanpa harus mengamati data terus-menerus.

10. UltramanDatabase.kt

Di file UltramanDatabase.kt, saya membuat kelas abstrak UltramanDatabase yang merupakan implementasi dari RoomDatabase, dan menjadi inti dari database lokal aplikasi

saya. Di sinilah saya mendefinisikan skema database dan menyediakan akses ke DAO (UltramanDao).

Anotasi @Database saya gunakan untuk memberitahu Room bahwa ini adalah database dan saya tentukan:

- entities = [UltramanItem::class], artinya hanya ada satu tabel, yaitu tabel ultraman yang berbasis dari data class UltramanItem.
- version = 3, yang berarti ini versi ketiga dari database. Kalau ada perubahan skema, versi ini harus ditingkatkan.
- exportSchema = false, karena saya tidak butuh Room menghasilkan file JSON skema.

Fungsi abstrak ultramanDao() akan diimplementasikan otomatis oleh Room, dan memungkinkan saya mengakses semua fungsi yang saya definisikan di UltramanDao.

Lalu di dalam companion object, saya buat singleton pattern agar hanya ada satu instance UltramanDatabase selama aplikasi berjalan. Saya gunakan @Volatile dan synchronized untuk memastikan instance ini thread-safe, penting agar tidak terjadi race condition ketika database pertama kali diakses dari berbagai thread.

Metode getDatabase(context: Context) memeriksa apakah instance sudah ada. Jika belum, saya buat dengan Room.databaseBuilder() menggunakan application context, lalu memberi nama database "ultraman_db". Saya tambahkan .fallbackToDestructiveMigration() supaya jika ada perbedaan versi database, Room akan mereset database (menghapus dan membuat ulang) daripada crash karena konflik migrasi.

11. UltramanRepository.kt

Di file UltramanRepository.kt, saya membuat kelas UltramanRepository yang berfungsi sebagai jembatan antara layer data (DAO dan Retrofit) dengan ViewModel. Ini adalah tempat saya menyatukan logika pengambilan data dari jaringan (TMDB API) dan penyimpanan lokal menggunakan Room. Dengan cara ini, ViewModel saya tetap bersih dan hanya fokus pada penyajian data.

Pertama, saya mendeklarasikan dua properti:

- ultramanItems: mengambil semua data dari database lokal dalam bentuk Flow, supaya UI bisa menampilkan data secara real-time.
- favoriteItems: mengambil semua item favorit, juga sebagai Flow, untuk ditampilkan di halaman favorit.

Fungsi refreshData(apiKey: String) adalah fungsi inti yang saya pakai untuk memuat data terbaru dari TMDB. Saya buat dalam bentuk Flow<ApiResponse<...>>, sehingga ViewModel dan UI bisa merespons statusnya (Loading, Success, atau Error).

Dalam fungsi tersebut, saya memulai dengan emit(ApiResponse.Loading) agar UI tahu sedang ada proses pengambilan data. Lalu saya memanggil searchUltraman() dari Retrofit, yang akan mengembalikan hasil pencarian Ultraman dari TMDB. Hasil results saya konversi satu per satu ke UltramanItem melalui ekstensi toUltramanItem().

Setelah itu, saya ambil data lama dari database menggunakan getAllOnce() untuk mengecek status favorit sebelumnya. Ini penting karena data dari API tidak menyimpan informasi favorit. Jadi saya cocokkan data baru dengan data lama, dan kalau id cocok, saya salin nilai isFavorite agar tidak hilang.

Saya kemudian simpan data baru ke database dengan dao.insertAll(...), dibatasi hanya 20 item pertama agar tidak membebani aplikasi. Terakhir, saya emit hasil sukses berupa ApiResponse.Success(...).

Kalau ada error saat proses, misalnya koneksi internet gagal, saya tangani dengan catch, log error-nya, dan emit ApiResponse.Error.

Selain itu, saya juga buat fungsi toggleFavorite() untuk membalik status favorit sebuah item berdasarkan id dan nilai saat ini. Fungsi ini dipanggil dari UI ketika pengguna menekan ikon favorit.

Terakhir, fungsi getFavorites() hanya mengembalikan ulang dao.getFavorites(), disediakan agar ViewModel bisa mengaksesnya langsung bila dibutuhkan.

12. UltramanViewModel.kt

Pada file UltramanViewModel.kt, saya membangun ViewModel yang berfungsi sebagai penghubung antara UI dan UltramanRepository. Di sini, saya kelola data Ultraman secara

reaktif menggunakan StateFlow, agar UI bisa merespons perubahan data secara real-time dan tetap konsisten meskipun terjadi perubahan konfigurasi (seperti rotasi layar).

Pertama-tama, saya deklarasikan beberapa MutableStateFlow sebagai state internal:

- _ultramanList menyimpan daftar seluruh data Ultraman dari database lokal.
- _favoriteList menyimpan daftar yang telah ditandai favorit.
- _uiState menyimpan status API saat pengambilan data dari internet (Loading, Success, atau Error).
- _favoriteState menyimpan status pengambilan data favorit, dengan struktur respons yang sama seperti _uiState.

Untuk setiap MutableStateFlow, saya expose versinya yang StateFlow ke luar agar hanya bisa dibaca, bukan dimodifikasi langsung oleh UI.

Di dalam init, saya langsung memanggil refreshData() untuk mengambil data awal dari API TMDB, dan juga observeLocalData() agar bisa terus memantau data dari database secara live.

Dalam fungsi observeLocalData(), saya meluncurkan dua collectLatest coroutine:

- Pertama, untuk mengambil semua data dari repository.ultramanItems, lalu menyimpannya ke _ultramanList dan sekaligus memperbarui _uiState menjadi sukses.
- Kedua, untuk data favorit, saya lakukan hal serupa. Saya juga bungkus dengan try-catch agar bila terjadi error saat pengambilan data favorit, UI tetap mendapatkan ApiResponse.Error.

Fungsi refreshData() bisa dipanggil dari UI untuk memaksa ambil data terbaru dari TMDB. Ia memanggil fungsi dari repository dan mengalirkan hasilnya langsung ke _uiState.

Untuk kebutuhan seperti menampilkan detail, saya sediakan fungsi getItemById(id) yang mencari satu item berdasarkan ID dari daftar yang sudah dimuat.

Terakhir, untuk menangani aksi favorit dari pengguna, saya buat fungsi toggleFavorite(item). Fungsi ini akan memanggil repository agar membalik nilai isFavorite di database. Karena perubahan disimpan ke database, UI akan otomatis merespons berkat collectLatest yang berjalan di awal tadi.

13. UltramanViewModelFactory.kt

Pada file UltramanViewModelFactory.kt, saya membuat sebuah ViewModelProvider.Factory yang digunakan untuk menciptakan instance dari UltramanViewModel dengan dependensi yang tidak bisa disediakan secara default oleh Android (yaitu Repository yang memerlukan Dao, dan Dao yang butuh Context untuk mengakses Room database).

Konstruktor UltramanViewModelFactory menerima parameter Context, yang akan digunakan untuk menginisialisasi UltramanDatabase. Di dalam fungsi create(), saya panggil UltramanDatabase.getDatabase(context) untuk mendapatkan instance Room database, lalu saya ambil ultramanDao()-nya.

Dengan Dao tersebut, saya buat instance UltramanRepository, dan dari repository ini saya akhirnya membentuk instance UltramanViewModel.

Karena fungsi create() mengembalikan tipe generik T, saya perlu melakukan casting ke T dengan anotasi @Suppress("UNCHECKED_CAST") agar tidak terjadi error saat kompilasi. Ini adalah pendekatan standar saat menggunakan ViewModel yang membutuhkan konstruktor khusus, dan memastikan integrasi dengan ViewModelProvider berjalan lancar saat membuat ViewModel di dalam Activity atau Composable.

14. ListScreen.kt

Pada file ListScreen.kt, saya membuat tampilan utama yang menampilkan daftar seluruh data Ultraman dari API dan database lokal. Komponen utama yang saya gunakan adalah LazyColumn untuk menampilkan setiap item dalam bentuk kartu. Di dalam setiap kartu, saya tampilkan poster Ultraman dengan GlideImage, judul, tahun rilis, deskripsi singkat, dan dua tombol: satu untuk membuka tautan TMDB di browser (Intent.ACTION_VIEW) dan satu lagi untuk navigasi ke halaman detail dengan navController.

Saya juga menambahkan tombol favorit berupa ikon hati di pojok kiri atas gambar poster. Tombol ini bisa diklik untuk mengubah status favorit item tersebut melalui fungsi viewModel.toggleFavorite(item). Untuk menjaga tampilan tetap responsif dan aman dari error, saya membungkus state data dengan ApiResponse yang memiliki tiga kondisi: Loading, Error, dan Success. Jika data sedang dimuat, akan muncul

CircularProgressIndicator. Jika terjadi error, muncul pesan error. Dan jika sukses, data ditampilkan dalam daftar.

15. DetailScreen.kt

File DetailScreen.kt berfungsi untuk menampilkan detail dari salah satu item Ultraman. Data detail ini saya ambil berdasarkan itemId yang dikirim dari halaman sebelumnya melalui navigasi. Saya memanggil fungsi viewModel.getItemById(itemId) untuk mengambil objek Ultraman yang sesuai. Jika data ditemukan, saya tampilkan poster dalam ukuran besar di bagian atas dengan GlideImage, lalu dilanjutkan dengan judul, tahun rilis, dan sinopsis atau deskripsi lengkapnya. Semua informasi saya bungkus dalam Column yang bisa discroll menggunakan verticalScroll, agar tampilan tetap rapi di berbagai ukuran layar.

Jika itemId tidak valid atau data tidak ditemukan, maka saya tampilkan teks "Data tidak ditemukan" di tengah layar sebagai fallback.

16. FavoriteScreen.kt

FavoriteScreen.kt memiliki struktur yang hampir sama dengan ListScreen, tapi hanya menampilkan data yang telah ditandai sebagai favorit oleh pengguna. Saya menggunakan state viewModel.favoriteState untuk mengelola datanya, yang juga dibungkus dengan ApiResponse agar dapat menangani loading, error, maupun data kosong dengan baik. Jika tidak ada item favorit, saya tampilkan pesan abu-abu di tengah layar agar pengguna tahu bahwa belum ada data yang difavoritkan.

Setiap item favorit ditampilkan dalam bentuk kartu, sama seperti di ListScreen, lengkap dengan gambar poster, judul, tahun, dan deskripsi singkat. Saya juga menyertakan dua tombol: "More Info" untuk membuka link TMDB, dan "Detail" untuk navigasi ke halaman detail. Ikon favorit di pojok kiri atas gambar tetap tersedia agar pengguna bisa menghapus item dari favorit langsung dari layar ini.

17. GlideImage.kt

Pada file GlideImage.kt, saya membuat sebuah komponen @Composable khusus bernama GlideImage yang fungsinya adalah untuk menampilkan gambar menggunakan Glide, library populer dari Android untuk memuat gambar dari internet. Karena Glide sendiri tidak bisa digunakan langsung di Jetpack Compose (karena dia berbasis View), maka saya gunakan AndroidView untuk menyisipkan ImageView ke dalam dunia Compose.

Saya menerima beberapa parameter: imageUrl untuk URL gambar, contentDescription untuk aksesibilitas, modifier untuk styling Compose seperti ukuran dan padding, serta contentScale untuk mengatur cara gambar di-scale. Nilai contentScale ini saya konversikan ke ImageView.ScaleType supaya tetap konsisten antara Compose dan View klasik.

Di dalam AndroidView, saya buat ImageView baru dan atur scaleType-nya berdasarkan contentScale. Kemudian di bagian update, saya panggil Glide.with(context).load(imageUrl).into(it) untuk memuat gambar ke ImageView tersebut. Dengan cara ini, saya bisa tetap menggunakan kemampuan Glide dalam proyek Compose tanpa harus migrasi ke library lain seperti Coil.

18. BottomBarNavigation.kt

Pada file ini, saya membuat komponen navigasi bawah (BottomBarNavigation) menggunakan Material 3 di Jetpack Compose. Tujuannya adalah agar pengguna bisa berpindah antar layar utama, yaitu "Daftar" dan "Favorit", dengan mudah melalui bottom bar.

Pertama, saya mendefinisikan sebuah data class bernama BottomNavItem yang menyimpan informasi dasar setiap item navigasi: judul (title), ikon (icon), dan rute (route) yang akan digunakan untuk navigasi. Kemudian, saya membuat daftar item bottom bar dalam variabel bottomNavItems, di mana "Daftar" menggunakan ikon film (Icons.Default.Movie) dan rutenya "list", sementara "Favorit" menggunakan ikon hati (Icons.Default.Favorite) dengan rute "favorite".

Dalam fungsi BottomBarNavigation, saya menggunakan NavigationBar dari Material 3. Di dalamnya, saya ambil rute aktif saat ini dari navController.currentBackStackEntryAsState() untuk menentukan item mana yang sedang aktif (dipilih). Lalu, saya melakukan iterasi pada bottomNavItems dan menampilkan masing-masing sebagai NavigationBarItem.

Setiap item akan dianggap terpilih (selected) jika currentRoute cocok dengan item.route. Saat pengguna mengetuk item, saya melakukan navigasi ke rute tersebut hanya jika belum berada di rute yang sama, agar tidak membuat tumpukan navigasi menumpuk. Saya juga menggunakan popUpTo("list") agar kembali ke awal saat perlu, dan launchSingleTop = true agar tidak membuat layar ganda jika sudah berada di rute itu.

Dengan pendekatan ini, saya memastikan bottom bar tetap sederhana, responsif, dan intuitif untuk berpindah antar halaman utama aplikasi.

TAUTAN GIT

Berikut adalah tautan untuk semua source code yang telah dibuat.

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