LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Natalie Grace Katiandagho **NIM. 2310817120003**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

Laporan Praktikum Pemrograman Mobile
Modul 1: Android Basic With Kotlin
Modul 2: Android Layout
Modul 3: Build A Scrollable List
Modul 4: View Model and Debugging
Modul 5: Connect to the Internet
ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile.
Laporan Akhir Praktikum ini dikerjakan oleh:

Nama Praktikan : Natalie Grace Katiandagho
NIM : 2310817120003

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

DAFTAR ISI

DAFTAR GAMBAR

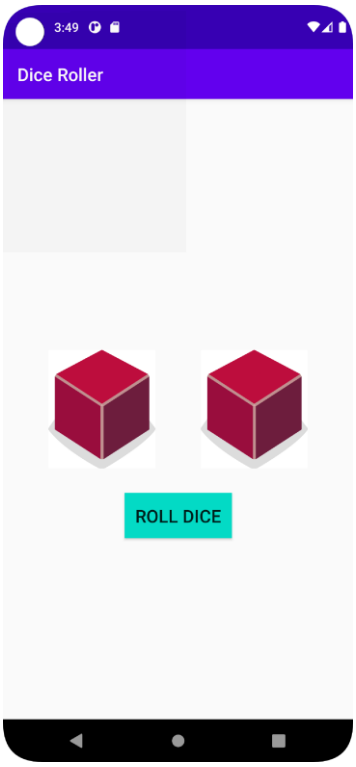
DAFTAR TABEL

MODUL 1 : Android Basic With Kotlin

SOAL 1

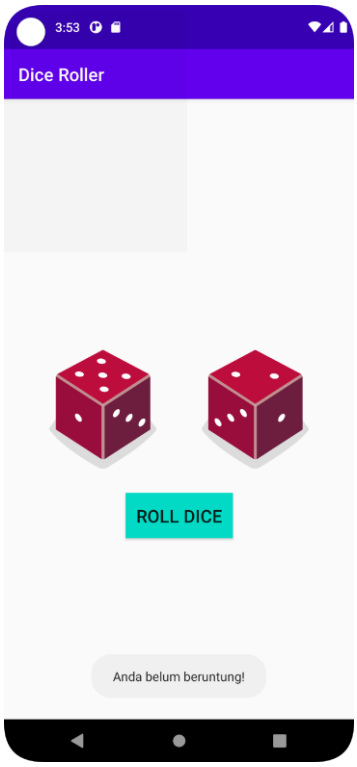
Soal Praktikum:
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol "Roll Dice". Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



Gambar 1. Tampilan Awal Aplikasi

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.

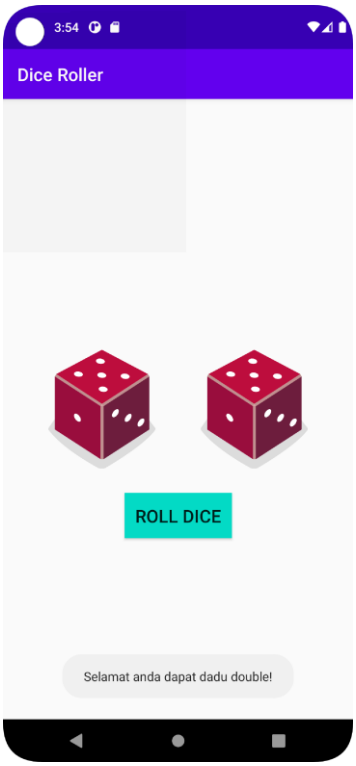


Gambar 2. Tampilan Dadu Setelah Di Roll

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

A. Source Code

Tabel 1. MainActivity Modul1

```
1  package com.example.xml_diceroller
2
3  import android.os.Bundle
   import android.widget.Toast.LENGTH_LONG
   import androidx.activity.enableEdgeToEdge
   import androidx.appcompat.app.AppCompatActivity
   import androidx.core.view.ViewCompat
   import androidx.core.view.WindowInsetsCompat
   import
   com.example.xml_diceroller.databinding.ActivityMainBinding
   import com.google.android.material.snackbar.Snackbar

   class MainActivity : AppCompatActivity() {
       private lateinit var binding : ActivityMainBinding
       override fun onCreate(savedInstanceState: Bundle?) {
           super.onCreate(savedInstanceState)
           enableEdgeToEdge()
           binding =
```

```
ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.button.setOnClickListener {
            NgerollDadu()
        }
    }
    fun NgerollDadu() {
        val dadu = dadu(6)
        val sisi = dadu.NgerollDadu()
        val sisi2 = dadu.NgerollDadu()
        val NgerollDadu1 = when (sisi) {
            1 -> R.drawable.dadu1
            2 -> R.drawable.dadu2
            3 -> R.drawable.dadu3
            4 -> R.drawable.dadu4
            5 -> R.drawable.dadu5
            else -> R.drawable.dadu6
        }
        val NgerollDadu2 = when (sisi2) {
            1 -> R.drawable.dadu1
            2 -> R.drawable.dadu2
            3 -> R.drawable.dadu3
            4 -> R.drawable.dadu4
            5 -> R.drawable.dadu5
            else -> R.drawable.dadu6
        }

        binding.sisi.setImageResource(NgerollDadu1)
        binding.sisi2.setImageResource(NgerollDadu2)

        val pesan = if (sisi == sisi2) {
            "Selamat, anda mendapatkan dadu double"
        } else {
            "Anda belum beruntung"
        }
        Snackbar.make(binding.button, pesan,
LENGTH_LONG).show()
    }
}

class dadu(private val numSisi : Int)
{
```

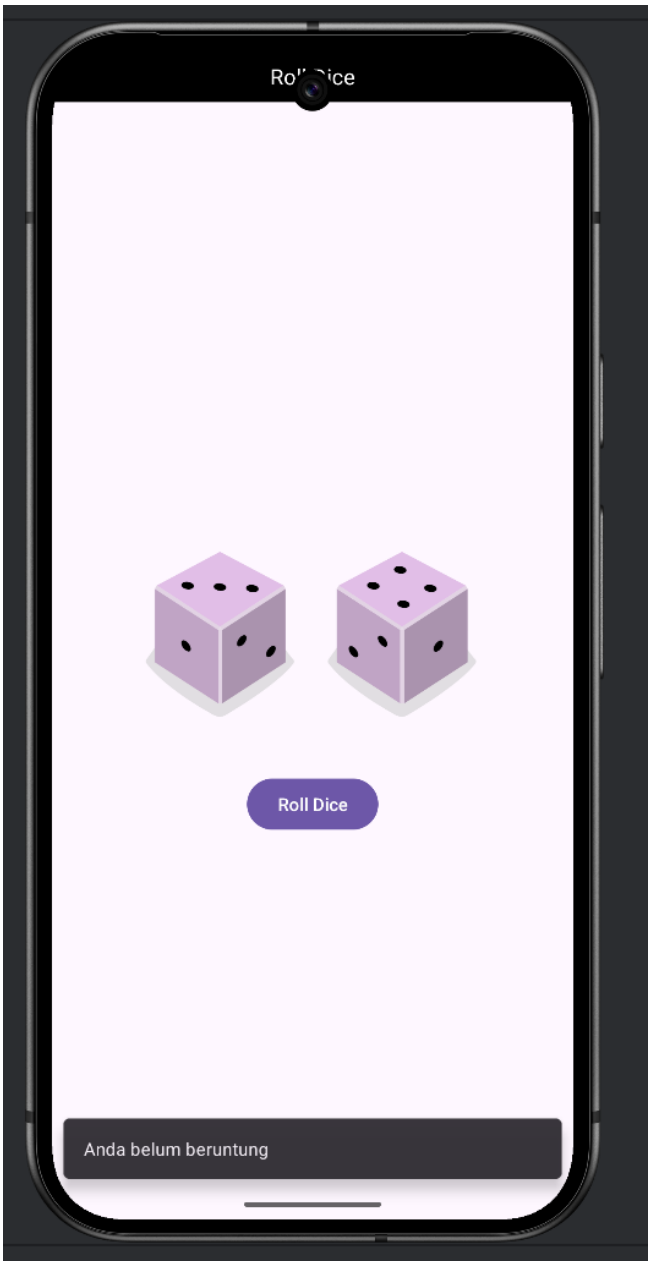
```
    fun NgerollDadu() : Int
    {
        return(1..numSisi).random()
    }
}
```

Tabel 2. ActivityMain Modul1

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="42dp"
        android:background="@color/black"
        android:paddingVertical="10dp"
        android:text="Roll Dice"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="17dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        />

    <ImageView
        android:id="@+id/sisi"
        android:layout_width="150dp"
        android:layout_height="170dp"
        android:src="@drawable/dadu0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintHorizontal_bias="0.22"
        app:layout_constraintVertical_bias="0.5"/>

    <ImageView
        android:id="@+id/sisi2"
        android:layout_width="150dp"
        android:layout_height="170dp"
        android:src="@drawable/dadu0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_bias="0.77"
        app:layout_constraintVertical_bias="0.5"  />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:layout_editor_absoluteX="-40dp" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:text="Roll Dice"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/sisi"/>

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

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

1. MainActivity.kt

2. ActivityMain.kt

MODUL 2 : Android Layout

SOAL 1

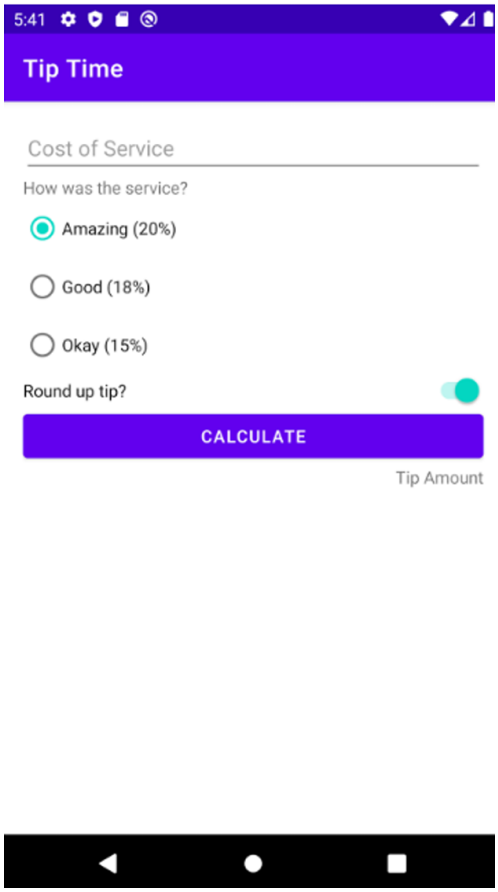
Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

1. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.

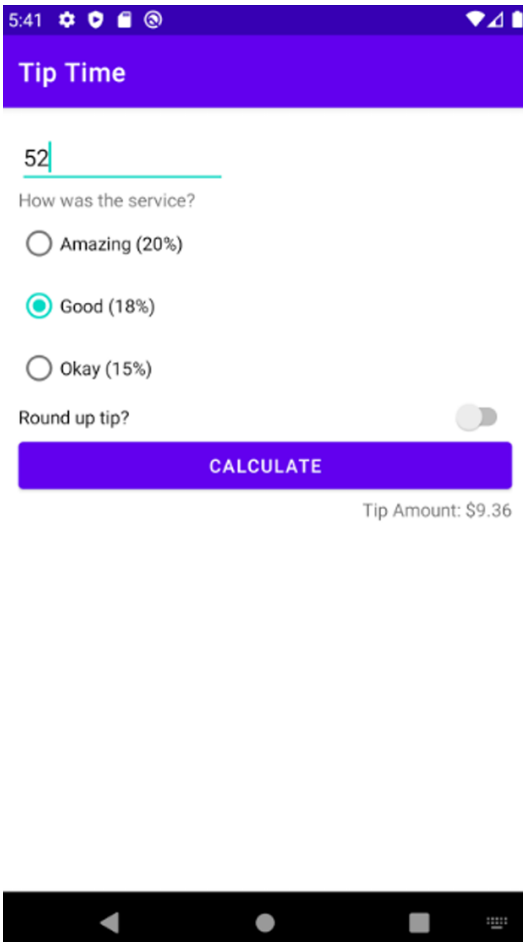
2. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.

3. Pengaturan Pembulatan Tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.

4. Tampilan Hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 5. Tampilan Awal Aplikasi



Gambar 6. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

Tabel 3. MainActivity Modul 2

```
1   package com.example.tipcalculator
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.annotation.StringRes
7   import androidx.compose.foundation.layout.Arrangement
8   import androidx.compose.foundation.layout.Column
9   import androidx.compose.foundation.layout.Row
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.rememberScrollState
15  import androidx.compose.foundation.text.KeyboardOptions
```

```
16  import androidx.compose.foundation.verticalScroll
17  import androidx.compose.material3.MaterialTheme
18  import androidx.compose.material3.RadioButton
19  import androidx.compose.material3.Surface
20  import androidx.compose.material3.Switch
21  import androidx.compose.material3.Text
22  import androidx.compose.material3.TextField
23  import androidx.compose.runtime.Composable
24  import androidx.compose.runtime.getValue
25  import androidx.compose.runtime.mutableDoubleStateOf
26  import androidx.compose.runtime.mutableStateOf
27  import androidx.compose.runtime.remember
28  import androidx.compose.runtime.setValue
29  import androidx.compose.ui.Alignment
30  import androidx.compose.ui.Modifier
31  import androidx.compose.ui.res.stringResource
32  import androidx.compose.ui.text.input.ImeAction
33  import androidx.compose.ui.text.input.KeyboardType
34  import androidx.compose.ui.tooling.preview.Preview
35  import androidx.compose.ui.unit.dp
36  import com.example.tipcalculator.ui.theme.TipCalculatorTheme
37  import java.text.NumberFormat
38  import kotlin.math.ceil
39
40  class MainActivity : ComponentActivity() {
41      override fun onCreate(savedInstanceState: Bundle?) {
42          super.onCreate(savedInstanceState)
43          setContent {
44              TipCalculatorTheme {
45                  Surface(modifier = Modifier.fillMaxSize()) {
46                      TipCalculatorLayout()
47                  }
48              }
49          }
50      }
51  }
52
53  @Composable
54  fun TipCalculatorLayout() {
55      var amountInput by remember { mutableStateOf("") }
56      var tipPercent by remember { mutableDoubleStateOf(15.0) }
57      var roundUp by remember { mutableStateOf(false) }
58
59      val amount = amountInput.toDoubleOrNull() ?: 0.0
60      val tip = calculateTip(amount, tipPercent, roundUp)
61
62      Column(
63          modifier = Modifier
64              .padding(32.dp)
```

```
65              .verticalScroll(rememberScrollState()),
66          horizontalAlignment = Alignment.CenterHorizontally,
67          verticalArrangement = Arrangement.spacedBy(16.dp)
68      ) {
69          Text(text = stringResource(R.string.calculate_tip),
    style = MaterialTheme.typography.headlineMedium)
70
71          EditNumberField(
72              label = R.string.bill_amount,
73              value = amountInput,
74              onValueChanged = { amountInput = it },
75              keyboardOptions = KeyboardOptions(
76                  keyboardType = KeyboardType.Number,
77                  imeAction = ImeAction.Next
78              )
79          )
80
81          TipOptions(tipPercent) {
82              tipPercent = it
83          }
84
85          RoundTheTipRow(roundUp = roundUp, onRoundUpChanged = {
    roundUp = it })
86
87          Text(
88              text = stringResource(R.string.tip_amount, tip),
89              style = MaterialTheme.typography.headlineSmall
90          )
91      }
92  }
93
94  @Composable
95  fun EditNumberField(
96      value: String,
97      @StringRes label: Int,
98      onValueChanged: (String) -> Unit,
99      keyboardOptions: KeyboardOptions,
100     modifier: Modifier = Modifier
101 ) {
102     TextField(
103         value = value,
104         onValueChange = onValueChanged,
105         singleLine = true,
106         modifier = modifier.fillMaxWidth(),
107         label = { Text(stringResource(label)) },
108         keyboardOptions = keyboardOptions
109     )
110 }
111
```

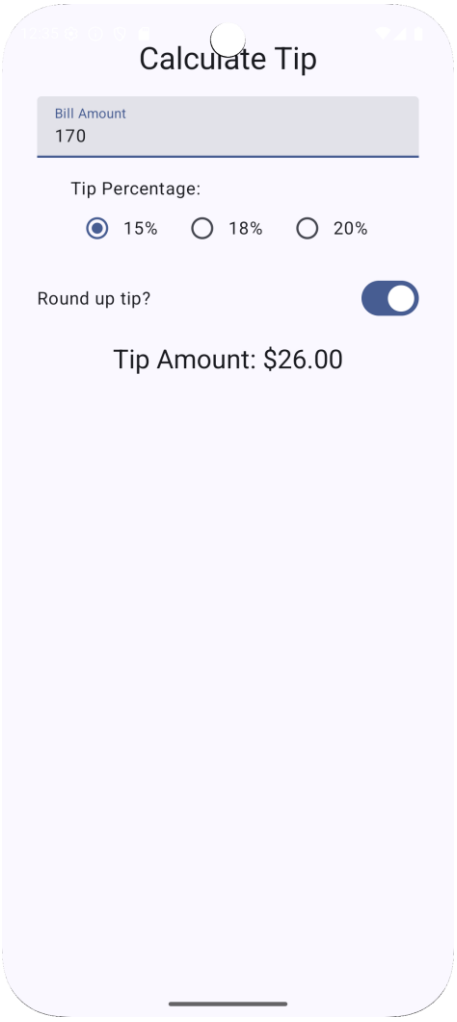
```
@Composable
fun TipOptions(selectedTip: Double, onTipSelected: (Double) ->
Unit) {
    Column {
        Text(text = "Tip Percentage:")
        Row(verticalAlignment = Alignment.CenterVertically) {
            listOf(15.0, 18.0, 20.0).forEach { percent ->
                Row(
                    verticalAlignment =
Alignment.CenterVertically,
                    modifier = Modifier.padding(end = 16.dp)
                ) {
                    RadioButton(
                        selected = selectedTip == percent,
                        onClick = { onTipSelected(percent) }
                    )
                    Text(text = "${percent.toInt()}%")
                }
            }
        }
    }
}

@Composable
fun RoundTheTipRow(
    roundUp: Boolean,
    onRoundUpChanged: (Boolean) -> Unit
) {
    Row(
        modifier = Modifier
            .fillMaxWidth()
            .height(48.dp),
        verticalAlignment = Alignment.CenterVertically,
        horizontalArrangement = Arrangement.SpaceBetween
    ) {
        Text(
            text = stringResource(R.string.round_up_tip),
            modifier = Modifier.weight(1f)
        )
        Switch(
            checked = roundUp,
            onCheckedChange = onRoundUpChanged
        )
    }
}

private fun calculateTip(amount: Double, tipPercent: Double =
15.0, roundUp: Boolean): String {
    var tip = tipPercent / 100 * amount
```

```
158     if (roundUp) {
159         tip = ceil(tip)
160     }
161     return NumberFormat.getCurrencyInstance().format(tip)
162 }
163 
164 @Preview(showBackground = true)
165 @Composable
166 fun TipCalculatorPreview() {
167     TipCalculatorTheme {
168         TipCalculatorLayout()
169     }
170 }
```

Tabel 4. String.xml Modul 2

```
1   <resources>
2       <string name="app_name">Tip Calculator</string>
3       <string name="calculate_tip">Calculate Tip</string>
4       <string name="bill_amount">Bill Amount</string>
5       <string name="tip">Tip (%)</string>
6       <string name="round_up_tip">Round up tip?</string>
7       <string name="tip_amount">Tip Amount: %1$s</string>
8   </resources>
```

B. Output Program



Gambar 7. Screenshoot Hasil Modul 2

C. Pembahasan

1. MainActivity.kt:

Pada line 1, dideklarasikan nama package file Kotlin yang pada praktikum kali ini adalah com.example.tipcalculcator

Dari line 3-38 fungsinya untuk mengimport :

Pada line 3, Bundle buat komunikasiin data antar aktivitas

Pada line 4, ComponentActivity sebagai superclass untuk activity yang menggunakan Jetpack Compose
Pada line 5, setContent untuk menampilkan UI berbasis Compose pada activity
Pada line 6, @StringRes jadi anotasi untuk resource ID bertipe string
Pada line 7–13, diimpor beberapa komponen layout :

- Arragement, untuk pengaturan posisi child dalam layout
- Column, Row untuk struktur tata letak vertikal dan horizontal
- fillMaxSize, fillMaxWidth, height, padding untuk mengatur ukuran dan margin komponen

Pada line 14, rememberScrollState untuk menyimpan dan mengingat posisi scroll
Pada line 15, diimport KeyboardOptions untuk mengatur jenis keyboard dan aksi input
Pada line 16, diimport verticalScroll untuk memungkinkan scroll secara vertikal pada layout
Pada line 17–22, diimpor komponen Material 3:

- MaterialTheme, untuk mengakses tema aplikasi
- RadioButton untuk pemilihan opsi
- Switch untuk tombol on off fitur
- Text menampilkan tulisan ke layar
- TextField, digunakan untuk mengambil input teks dari pengguna

Pada line 23–28, fitur state Compose seperti:

- @Composable, untuk menandai fungsi sebagai composable
- remember, mutableStateOf, mutableDoubleStateOf, untuk mendefinisikan dan mengingat state yang dapat berubah
- getValue, setValue, untuk properti delegasi state

Pada line 29–35 :

- Modifier untuk mengatur tampilan, ukuran, padding komponen
- stringResource untuk mengambil string dari resource menggunakan ID
- ImeAction, KeyboardType untuk pengaturan aksi keyboard dan tipe input
- @Preview untuk pratinjau UI
- dp untuk unit ukuran
- TipCalculatorTheme untuk menggunakan tema khusus aplikasi
- NumberFormat untuk format angka

Pada line 36, untuk menerapkan tema khusus aplikasi yang sudah didefinisikan di folder ui.theme.
Pada line 37, untuk format angka ke dalam format mata uang.
Pada line 38, ceil dari kotlin.math untuk membulatkan nilai ke atas.

Pada line 40, kelas MainActivity yang turunan dari ComponentActivity
Pada line 41, ada fungsi yang dipanggil saat activity pertama kali dibuat. Parameter savedInstanceState digunakan untuk menyimpan data jika activity perlu dibuat ulang.
Pada line 42, memanggil onCreate dari superclass (ComponentActivity) agar fungsi dasar activity tetap berjalan.
Pada line 43, setContent untuk menampilkan UI didalam kurung kurawalnya nanti jadi isi tampilan Activity.

Pada line 44, buat temanya
Pada line 45, suface buat tempat UI bisa dikasih warna dll, fillMaxSize buat ngisi layar penuh
Pada line 46, manggil fungsi TipCalculatorLayout

Pada line 53, buat nandain fungsi UI.
Pada line 54, buat fungsi utama tampilan kalkulator.
Pada line 55, nyimpan input tagihan dari pengguna.
Pada line 56, buat nyimpan presentase tipnya.
Pada line 57, buat nyimpan pilihan tip mau dibultkan atau engga.
Pada line 59, buat ngubah input ke angka, kalau kosong maka 0.0
Pada line 60, ngitung jumlah tip dari input, persen sama pembulatan tadi.

Pada line 62, buat nyusun semua elemen dari atas ke bawah.
Pada line 63 – 65, buat beri jarak 32dp di isi, terus buat kolom supaya bisa dicroll kalau isinya kepanjangan.
Pada line 66, buat rata tengah horizontal.
Pada line 67, buat ngasih jarak 16dp antar elemen.

Pada line 69, buat ngasih judul : Calculate Tip, uk medium.

Pada line 71-77, buat nampilin input angka di bill amount ada ambil nilai input, diperbarui setiap pengguna mengetik.

Pada line 81-82 buat nampilin pilihan persen tip

Pada line 85 isinya menampilkan switch on off buat tip dibulatkan atau engga, terus ngumbah nilai roundUp nya pas diganti.

Pada line 86-88, ada text lagi buat nampilkan perhitungan tip dalam bentuk teks – uk teks kecil.

Pada line 93 ada composable lagi buat tampila UI.
Pada line 94, buat nampilin input angka .
Pada line, 95-99 ada :

- value: String , Nilai input yang sedang diketik.
- label: Int (@StringRes) , ID dari teks label
- onValueChanged: (String) -> Unit fungsi saat pengguna mengetik untuk memperbarui nilai.
- keyboardOptions: KeyboardOptions ngatur jenis keyboard.
- modifier: Modifier untuk pengaturan tampilan tambahan.

Pada TextField (line 101-107) nampilin input teks satu baris, lebarmya fillMaxWidth, label teks dari stringResouce(label), keyboardnya sesuai keyboardOptions.

Pada 111 ada composable lagi yang merupakan fungsi UI.
Pada line 112, nampilkan pilihan persen
Pada 113, text buat label teks
Pada line 114, ada row buat ngasih pilihan secara horizontal, rata tengah vertical.
Pada line 115, buat isi tiga pilihan persen

Di line 116-123, ada setiap pilihan di dalam row, isinya ada tombol pilihan radiobutton, aktif nilainya sama selectedTip, pas di klik muncul onTipSelected(percent), text nya buat nampilin nilai persennya.

Pada line 130, sama seperti sebelumnya
Pada line 131-134, buat ngasih fungsi isinya teks sama switch untuk aktif atau nonaktif opsi dari pembulatan
Pada line 135-141, ada row buat elemen secara horizontal, arrangement.SpaceBetween buat jarak maz antara text dan switch, Alignment.CenterVertically buat mensejajarkan elemen secara vertical Tengah.
Pada line 142-145, buat nampilin text , Modifier.weight(1f) buat ngisi ruang yang tersedia.
Pada line 146-148, buat switch on off checked buat status switch ngikutin nilai roundUp, onCheckedChange = onRoundChanged buat manggil fungsi kalau switchnya diganti.

Pada line 153, merupakan fungsi ngitung jumlah tip berdasarkan input.

- amount: Double → Jumlah tagihan.
- tipPercent: Double (default 15.0) → Persentase tip.
- roundUp: Boolean → Jika true, hasil tip akan dibulatkan ke atas.

Isinya di line 155-158 :

- Hitung tip: tip = tipPercent / 100 * amount
- Jika roundUp == true, bulatkan ke atas dengan ceil(tip)
- Format hasil jadi bentuk uang: NumberFormat.getCurrencyInstance().format(tip)

Pada line 161-167, buat pratinjau tampilannya

- @Preview(showBackground = true) → Menampilkan preview dengan latar belakang.
- Di dalamnya, memanggil TipCalculatorLayout() dengan tema TipCalculatorTheme

2. **Strings.xml**

Pada baris 1, deklarasi resources yang berisi kumpulan string yang digunakan di seluruh aplikasi.
Pada baris 2, berisi nama aplikasi "Tip Calculator" yang akan muncul di layar utama aplikasi.
Pada baris 3, bernama calculate_tip untuk judul proses menghitung tip.
Pada baris 4, berisi teks "Bill Amount", digunakan untuk label input jumlah tagihan.
Pada baris 5, berfungsi sebagai label untuk memilih persentase tip, berisi teks "Tip (%)".
Pada baris 6, berisi teks "Round up tip?", digunakan sebagai label untuk opsi apakah ingin membulatkan tip atau tidak.
Pada baris 7, berisi teks "Tip Amount: %1$s", di mana %1$s adalah placeholder yang akan diisi dengan jumlah tip yang dihitung saat aplikasi dijalankan.
Pada baris 8, resource ditutup

MODUL 3 : Build A Scrollable List

SOAL 1

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

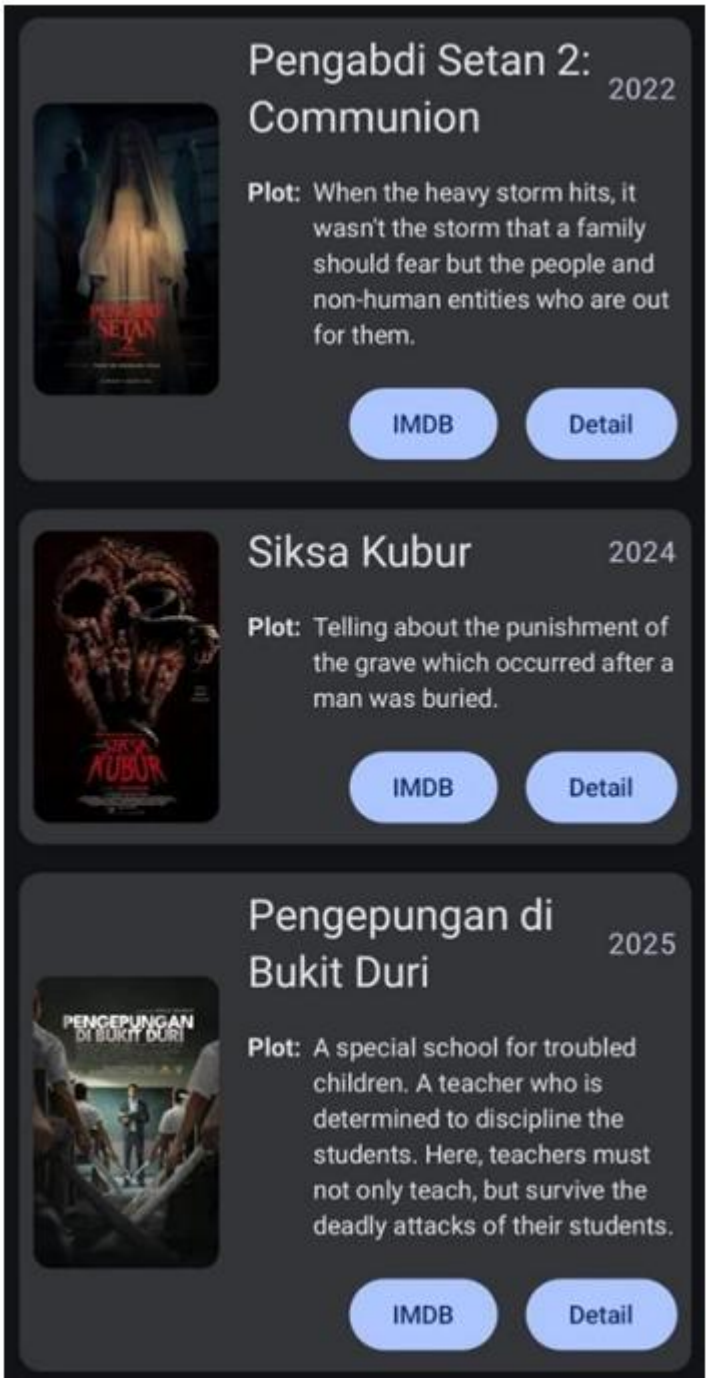
1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas
3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah
4. Terdapat 2 button dalam list, dengan fungsi berikut:
    a. Button pertama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

SOAL 2

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

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



Gambar 8.Contoh UI 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 9. Contoh UI Modul 3

Jawaban 1.

A. Source Code

1. MainActivity.kt

Tabel 5. MainActivity/kt Modul3

```
1  package com.example.gracemanhwa_picks
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.navigation.NavType
7  import androidx.navigation.compose.*
8  import androidx.navigation.navArgument
9  import com.example.gracemanhwa_picks.ui.theme.GracemanhwaPicksTheme
10
11 data class Manhwa(
12     val id: Int,
13     val title: String,
14     val author: String,
15     val description: String,
16     val imageRes: Int,
17     val url: String
18 )
19
20 val manhwaList = listOf(
21     Manhwa(
22         1,
23         "Solo Leveling",
24         "Chu-Gong",
25         "In a world where hunters — human warriors who possess supernatural abilities
       — must battle deadly monsters to protect all mankind from certain annihilation, a
       notoriously weak hunter named Sung Jin-woo finds himself in a seemingly endless
       struggle for survival. "
26             + "One day, after narrowly surviving an overwhelmingly powerful double
       dungeon that nearly wipes out his entire party, a mysterious program called the
       System chooses him as its sole player and in turn, gives him the unique ability to
       level up in strength. "
27             + "This is something no other hunter is able to do, as a hunter's abilities are
       set once they awaken. Jinwoo then sets out on a journey as he fights against all kinds
       of enemies, both man and monster, to discover the secrets of the dungeons and the
       true source of his powers. "
28             + "He soon discovers that he has been chosen to inherit the position of
```

```
Shadow Monarch, essentially turning him into an immortal necromancer who has absolute rule over the dead. "
            + "He is the only Monarch who fights to save humanity, as the other Monarchs are all trying to kill him and wipe out the humans.",
        R.drawable.solo_leveling,
        "https://www.tappytoon.com/en/book/solo-leveling-official"
    ),
    Manhwa(
        2,
        "Omniscient Reader",
        "SingNSong",
        "Kim Dokja is a young man leading a simple life, who has been the sole reader of a novel \"Three Ways to Survive in a Ruined World\" for 13 years of his life. "
            + "As he was reading the novel's final chapter, reality and the world of fiction started to merge, allowing him to appear at the beginning point of the story. "
            + "Being the only person who knew how the world could end, Kim Dokja is determined to create a different ending by solving and conquering various challenges, known as scenarios, which are operated by dokkaebi.",
        R.drawable.omniscient_reader,
        "https://www.webtoons.com/en/action/omniscient-reader/list?title_no=2154"
    ),
    Manhwa(
        3,
        "The Beginning After the End",
        "TurtleMe",
        "It follows the life of the late King Grey after his untimely and mysterious death. Reborn as Arthur Leywin, he seeks to correct his past mistakes in the vibrant new continent of Dicathen, a world of magic and fantastical creatures. "
            + "Equipped with the knowledge of a powerful king in his mid-thirties, Arthur navigates his new life as the magic-wielding child of two retired adventurers and gains purpose through each of his new experiences—something he lacked in his previous life. "
            + "When a kind dragon sacrifices her life to protect him, Arthur resolves to live a sincere, kind, and courageous life with those he loves. With the help of a lost elf princess and the Elven Kingdom of Elenoir, Arthur begins his long journey to find his true place in the world.\n\n"
            + "As the years pass, Arthur becomes more and more comfortable in this world, positioning himself as a young, but respected figure. However, deja-vu strikes as a war brews between Dicathen and the Vritra, a clan of banished deities now ruling over a faraway continent. "
            + "Arthur must rise as a leader, despite his fear of becoming the war-hardened monster he once was in his past life. "
            + "As the war rages on, Arthur discovers that he was not reborn to this world by chance...nor was he the only one.",
```

```
53          R.drawable.tbate,
54          "https://tapas.io/series/tbate-comic/info"
55      ),
56      Manhwa(
57          4,
58          "Eleceed",
59          "Jeho Son",
60          "Jiwoo is a kind-hearted young man who harnesses the lightning-quick reflexes
        of a cat to secretly make the world a better place – one saved little child or foster pet
        at a time. "
61                  + "Kayden is a secret agent on the run, who finds himself stuck in the body
        of a…um…decidedly fat old fluffy cat. "
62                  + "Together, armed with Jiwoo's superpowers and Kayden's uber-smarts,
        they're out to fight those forces who would let evil rule this world. "
63                  + "That is, if they can stand each other long enough to get the job done.",
64          R.drawable.eleceed,
65          "https://www.webtoons.com/en/action/eleceed/list?title_no=1571"
66      ),
67      Manhwa(
68          5,
69          "Killer Peter",
70          "Kim Junghyun",
71          "On the surface, Glory Hound is a simple human rights organization. In reality,
        the organization has some of the best assassins in the world, in charge of performing
        legendary.\n\n"
72                  + "One of their best members was simply known as Apostle Peter, and he
        retired in protest of the new leader, Raphael. However, resignations were not
        accepted, and Peter was soon ambushed. Despite his best efforts, he dies.\n\n"
73                  + "Instead of dying, though, Peter miraculously found himself back in his
        teenage body. He doesn't know why, but he knows one thing: he will destroy Glory
        Hound.",
74          R.drawable.killer_peter,
75          "https://www.webtoons.com/en/action/killer-peter/list?title_no=5816"
76      ),
77      Manhwa(
78          6,
79          "Player Who Can't Level Up",
80          "GaVinGe",
81          "When Kim Kigyu received his invitation to become a player (a unique-ability
        player, at that), he thought his struggles were over. But no matter how hard he tries,
        he just can't seem to get past level 1! "
82                  + "After five years of working as a guide on the lower floors of the tower,
        he's finally discovered his ability to link with "Egos" and raise his stats. "
83                  + "As his new skills unlock adventures in unexplored gates, Kigyu gets his
```

```
        chance to defy expectations and show the world that rank isn’t everything.",
84          R.drawable.player_cant_level_up,
85          "https://tapas.io/episode/2414063"
86      ),
87      Manhwa(
88          7,
89          "SSS-Class Revival Hunter",
90          "Shinnoa",
91          "After the Tower suddenly appeared, individuals who wished to pursue their
        personal values began to inhabit it, coming to be called \"hunters.\" "
92              + "Everyone had their own goals, but only a chosen few were
        acknowledged and given powerful skills by the mysterious structure. "
93              + "Kim Gong-Ja, a weak F-Class hunter without any skills, is envious of
        those who were blessed by the Tower. "
94              + "Letting his jealousy overcome him one day, Gong-Ja abruptly receives a
        S-Class skill that allows him to copy a skill from someone else—after they kill
        him.\n\n"
95              + "Sooner than he likes, Gong-Ja gets to test his newly acquired ability on
        the legendary hunter known as the Flame Emperor. "
96              + "As he is dying, Gong-Ja learns the evil truth about the man he once
        admired the most. "
97              + "Receiving another potent skill that allows him to revive and go back in
        time by 24 hours, Gong-Ja devises a plan to travel 11 years into the past to eliminate
        the Flame Emperor and cement himself as the world's best hunter.",
98          R.drawable.sss_class_hunter,
99          "https://www.mangaread.org/manga/sss-class-suicide-hunter/"
100     )
101 )
102
103 class MainActivity : ComponentActivity() {
104     override fun onCreate(savedInstanceState: Bundle?) {
105         super.onCreate(savedInstanceState)
106         setContent {
107             GracemanhwaPicksTheme {
108                 val navController = rememberNavController()
109                 NavHost(navController, startDestination = "list") {
110                     composable("list") {
111                         ListScreen(navController)
112                     }
113                     composable(
114                         "detail/{id}",
115                         arguments = listOf(navArgument("id") { type = NavType.IntType })
116                     ) { backStackEntry ->
117                         val id = backStackEntry.arguments?.getInt("id") ?: 0
```

```
                    val item = manhwaList.first { it.id == id }
                    DetailScreen(item)
                }
            }
        }
    }
}
```

Tabel 10. Source Code Jawaban Soal 1

2. ListScreen.kt

```
package com.example.gracemanhwa_picks

import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
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.res.painterResource
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ListScreen(navController: NavController) {
    val context = LocalContext.current

    Scaffold(
        topBar = {
            CenterAlignedTopAppBar(
                title = {
                    Text(
                        text = "Grz's Manhwa Picks",
                        style = MaterialTheme.typography.titleSmall
                    )
```

```
33              }
34          )
35      }
36  ) { innerPadding ->
37      LazyColumn(
38          modifier = Modifier
39              .padding(innerPadding)
40              .fillMaxSize()
41              .padding(8.dp)
42      ) {
43          items(manhwaList) { item ->
44              Card(
45                  shape = RoundedCornerShape(20.dp),
46                  modifier = Modifier
47                      .padding(8.dp)
48                      .fillMaxWidth()
49              ) {
50                  Column(modifier = Modifier.padding(8.dp)) {
51                      Image(
52                          painter = painterResource(id = item.imageRes),
53                          contentDescription = item.title,
54                          contentScale = ContentScale.Crop,
55                          modifier = Modifier
56                              .fillMaxWidth()
57                              .height(180.dp)
58                              .clip(RoundedCornerShape(16.dp))
59                      )
60                      Spacer(modifier = Modifier.height(8.dp))
61                      Text(
62                          item.title,
63                          style = MaterialTheme.typography.titleLarge
64                      )
65                      Text(
66                          "By ${item.author}",
67                          style = MaterialTheme.typography.bodyMedium
68                      )
69                      Row(
70                          modifier = Modifier
71                              .fillMaxWidth()
72                              .padding(top = 8.dp),
73                          horizontalArrangement = Arrangement.SpaceEvenly
74                      ) {
75                          Button(onClick = {
76                              val intent = Intent(Intent.ACTION_VIEW, Uri.parse(item.url))
```

```
                        context.startActivity(intent)
                    }) {
                        Text("Baca")
                    }
                    Button(onClick = {
                        navController.navigate("detail/${item.id}")
                    }) {
                        Text("Detail")
                    }
                }
            }
        }
    }
}
```

Tabel 2. Source Code Jawaban Soal 1

3. DetailScreen.kt

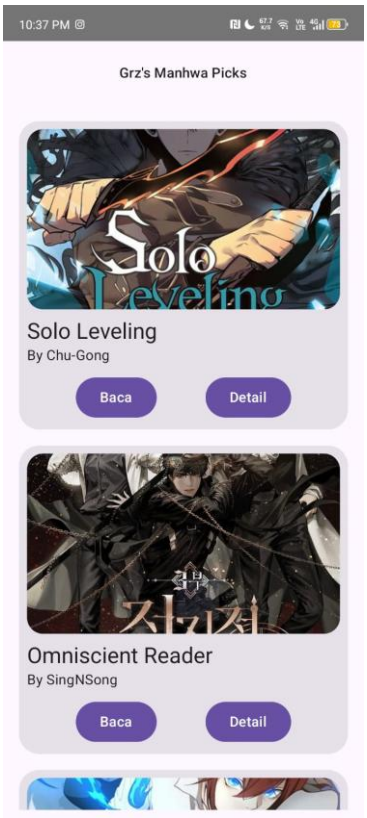
```
package com.example.gracemanhwa_picks

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.rememberScrollState
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.verticalScroll
import androidx.compose.material3.*
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.res.painterResource
import androidx.compose.ui.unit.dp

@Composable
fun DetailScreen(item: Manhwa) {
    val scrollState = rememberScrollState()

    Column(
        modifier = Modifier
            .fillMaxSize()
            .verticalScroll(scrollState)
            .padding(16.dp)
```

```
    ) {
        Image(
            painter = painterResource(id = item.imageRes),
            contentDescription = item.title,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
                .clip(RoundedCornerShape(16.dp)),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(item.title, style = MaterialTheme.typography.headlineMedium)
        Text("By ${item.author}", style =
MaterialTheme.typography.titleMedium)
        Spacer(modifier = Modifier.height(8.dp))
        Divider()
        Spacer(modifier = Modifier.height(8.dp))
        Text(item.description, style = MaterialTheme.typography.bodyLarge)
    }
}
```

4.Theme.kt

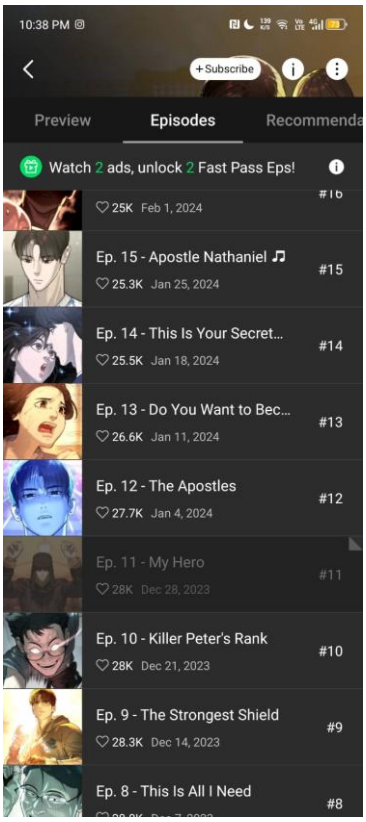
```
package com.example.gracemanhwa_picks.ui.theme

import androidx.compose.material3.*
import androidx.compose.runtime.Composable

@Composable
fun GracemanhwaPicksTheme(content: @Composable () -> Unit) {
    MaterialTheme(
        colorScheme = lightColorScheme(),
        typography = Typography(),
        content = content
    )
}
```

B. Output Program



Gambar 11. Output Modul 3

Gambar 12. Output Modul 3



Gambar 13. Output Modul 3

C. Pembahasan

1. MainActivity.kt:

Pada line 1, file ini berada di dalam package com.example.gracemanhwa_picks.
Pada line 3-9, mengimpor berbagai fungsionalitas untuk aplikasi:

- Line 3: Bundle untuk membawa data antar aktivitas.
- Line 4: ComponentActivity untuk aktivitas dengan Jetpack Compose.
- Line 5: setContent untuk menetapkan tampilan UI.
- Line 6: NavType untuk jenis data navigasi.
- Line 7: NavController untuk kontrol navigasi.
- Line 8: navArgument untuk mendefinisikan parameter navigasi.
- Line 9: GracemanhwaPicksTheme untuk tema aplikasi.

Pada line 11-18, didefinisikan sebuah data class bernama Manhwa. data class digunakan untuk membuat sebuah kelas yang hanya berfungsi untuk menyimpan data.

- val id: Int: Menyimpan ID manhwa, berupa bilangan bulat (Int).
- val title: String: Menyimpan judul manhwa, berupa teks (String).
- val author: String: Menyimpan nama pengarang manhwa, berupa teks (String).
- val description: String: Menyimpan deskripsi manhwa, berupa teks (String).

- val imageRes: Int: Menyimpan resource ID gambar yang digunakan untuk manhwa, berupa bilangan bulat (Int), biasanya menunjuk ke file gambar.
- val url: String: Menyimpan URL (tautan) untuk membaca manhwa, berupa teks (String).

Lalu selanjutnya ada val manhwa list yang ada manhwa() isinya list masing-masing property yang mencakup atau yang berisi : id, title, author, description, imageRes, url.

Begitu seterusnya lalu selanjutnya ada di line 103-119 :
Line 103: MainActivity adalah kelas utama aplikasi yang mewarisi dari ComponentActivity.
Line 104-105: onCreate dipanggil saat aktivitas dibuat, dan super.onCreate(savedInstanceState) memastikan aktivitas diinisialisasi dengan benar.
Line 106: setContent digunakan untuk menetapkan tampilan UI aplikasi dengan Jetpack Compose.
Line 107: GracemanhwaPicksTheme menetapkan tema aplikasi.
Line 108: navController dibuat untuk mengelola navigasi antar layar.
Line 109-110: NavHost mendefinisikan struktur navigasi, dengan layar awal "list".
Line 111-113: Menampilkan ListScreen saat navigasi ke "list".
Line 114-116: Menetapkan layar "detail/{id}" untuk menerima argumen id dan menavigasi ke detail.
Line 117-118: Mengambil id dari argumen navigasi dan memberi nilai default 0 jika tidak ada.
Line 119: Mencari item dari manhwaList yang memiliki id sesuai argumen.

2. **ListScreen.kt**

1 file berasa di package utama

3-18 buat import: ngatur halaman, ubah URL supaya bisa dibacam nampilin gambar, nyediakan layoutm komponen scrollable (kayak recyle view tpi ver compose), sudutnya agar bulat, import material 3, buat fungsi UI dengan @Composable, ngatur ukuran dan posisi dengan Modifier, potong tampilan jadi sudut bulat pakai clip, atur skala gambar biar pas di tempatnya, ambil context Android buat buka link, ambil gambar dari drawable pakai resource, atur ukuran elemen dengan satuan dp, dan terakhir, navigasi antar layar pakai NavController.

Line 20, kode ini akan menggunakan API eksperimental -- padahal belum stabil tapi supaya bisa dipakai makanya pakai ini
Line 21, ngasih tau list screen itu compoable
Line 22, definisi fungsi yang ada parameter dari navController buat navigasi layar di app
Line 23, buat dapetin contect yang dibutuhkan
Line 25, ngasih layout dasar struktur umum top bar, bottom bar, dan konten utama
Line 26, top bar ada center aligned

Line 27, judul posisi tengah.
Line 28-31, judul titlenya kecil

LazyColumn, yang memuat setiap item hanya saat dibutuhkan untuk efisiensi. Setiap item dibungkus dalam Card dengan sudut melengkung dan padding, yang berisi gambar, judul manhwa, nama pengarang, dan dua tombol. Gambar ditampilkan dengan efek crop agar sesuai dengan ukuran yang telah ditentukan dan memiliki sudut melengkung. Setelah gambar, terdapat teks yang menampilkan judul dan pengarang manhwa dengan gaya teks yang telah disesuaikan. Di bagian bawah, terdapat dua tombol: tombol pertama untuk membuka URL manhwa di browser menggunakan Intent, dan tombol kedua untuk menavigasi ke layar detail manhwa dengan NavController

3. **DetailScreen.kt**

Pada line 1 itu menunjukan kalau file ada di package utama pada aplikasi

Line 3-14 itu fungsinya untuk import :

- 3 nampilin gambar
- 4 layout dasar
- 5 ingat status scroll
- 6 sudut lengkung
- 7 column scroll vertikal
- 8 material design 3
- 9 nandain kalau composable
- 10 modifier -- tata letak, uk., padding, dll
- 11 motong bentuk tampilan
- 12 ngatur gambar ditampilkannya gimana
- 13 membuat gambar dari file lain
- 14 untuk ukuran supaya konsisten

16 buat nandain itu compose

Line 17 nampilin detail manhwa

18 supaya bisa scroll + diingat

20 isinya column buat nyusus komponen vertikal

21 fill max size berarti ukuran layarnya penuh

22 bisa di scroll ke bawah + diingat

23 jarak

26 image -- buat nampilin gambar

27-33 ngambil gambar dari file, ada deskripsi konten, di modifier ada lebar gambar yang ngikutin lebar layar, tingginya 250, gambarnya sudutnya dibulatkan, gambarnya dipastiin penuh

35 beri jarak vertikal 16dp per elemen

36 nampilim judul ukurannya medium

37 nampilin nama author tulisannya sedang juga

38 jarak setelah garis 8dp

39 divider buat garis horizontal tipis

40 jarak lagi setelah garis 8dp

41 eskripsi panjangnya

4. **Theme.kt**

Pada line 1 ada package untuk tema dari aplikasi
Pada line 3-4 digunakan untuk import elemen compose material 3 dan composable function
Pada line 6 ada composable -- untuk UI
Lalu line 7 untuk menerima composable lain buat parameternya, jadi UI nya dibungkus.
Pada line 8 ada material theme -- ini fungsi untuk tema
Line 9 ada skema warnanya di mode terang
Line 10 untuk mengatur gaya huruf
Line 11 terdapat content untuk menampilkan isi UI yang dibungkus

Jawaban nomor 2 .

Meskipun LazyColumn menawarkan kode yang lebih singkat dan deklaratif dibandingkan RecyclerView, ada beberapa alasan mengapa RecyclerView masih banyak digunakan dalam pengembangan Android. Pertama, banyak aplikasi yang sudah ada menggunakan RecyclerView, dan migrasi ke LazyColumn memerlukan upaya besar. Kedua, RecyclerView memberikan kontrol lebih besar dalam hal kustomisasi, seperti animasi item dan dekorasi, yang penting untuk aplikasi yang membutuhkan tampilan dan interaksi yang lebih spesifik. Selain itu, RecyclerView didukung oleh banyak pustaka pihak ketiga yang membantu mempercepat pengembangan dan menambahkan fungsionalitas tambahan. Pada kasus daftar yang sangat besar atau kompleks, RecyclerView sering kali memberikan kinerja yang lebih

baik. Terakhir, banyak pengembang Android yang sudah berpengalaman dengan RecyclerView dan lebih nyaman menggunakannya, terutama dalam proyek yang memerlukan kontrol detail. Meski LazyColumn lebih modern, RecyclerView tetap relevan, terutama dalam proyek besar atau yang membutuhkan kustomisasi tinggi.

MODUL 4 : View Model 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

A. Source Code

Tabel 6. Manhwa.kt Modul 4

```
1  package com.example.gracemanhwa_picks.data
2
3  data class Manhwa(
4      val id: Int,
5      val title: String,
6      val author: String,
7      val description: String,
8      val imageRes: Int,
```

```
    val url: String
)
```

Tabel 7.ManhwaRepository.kt

```
package com.example.gracemanhwa_picks.data

import com.example.gracemanhwa_picks.R

class ManhwaRepository {
    fun getManhwas(): List<Manhwa> {
        return listOf(
            Manhwa(
                id = 1,
                title = "Solo Leveling",
                author = "Chu-Gong",
                description = "In a world where hunters — human warriors who possess supernatural abilities — must battle deadly monsters to protect all mankind from certain annihilation, a notoriously weak hunter named Sung Jin-woo finds himself in a seemingly endless struggle for survival. "
                        + "One day, after narrowly surviving an overwhelmingly powerful double dungeon that nearly wipes out his entire party, a mysterious program called the System chooses him as its sole player and in turn, gives him the unique ability to level up in strength. "
                        + "This is something no other hunter is able to do, as a hunter's abilities are set once they awaken. Jinwoo then sets out on a journey as he fights against all kinds of enemies, both man and monster, to discover the secrets of the dungeons and the true source of his powers. "
                        + "He soon discovers that he has been chosen to inherit the position of Shadow Monarch, essentially turning him into an immortal necromancer who has absolute rule over the dead. "
                        + "He is the only Monarch who fights to save humanity, as the other Monarchs are all trying to kill him and wipe out the humans.",
```

```
17                  imageRes =
    R.drawable.solo_leveling,
18                  url =
    "https://www.tappytoon.com/en/book/solo-leveling-
    official"
19              ),
20              Manhwa(
21                  id = 2,
22                  title = "Omniscient Reader",
23                  author = "SingNSong",
24                  description = "Kim Dokja is a young
    man leading a simple life, who has been the sole
    reader of a novel \"Three Ways to Survive in a
    Ruined World\" for 13 years of his life. "
25                          + "As he was reading the
    novel's final chapter, reality and the world of
    fiction started to merge, allowing him to appear at
    the beginning point of the story. "
26                          + "Being the only person
    who knew how the world could end, Kim Dokja is
    determined to create a different ending by solving
    and conquering various challenges, known as
    scenarios, which are operated by dokkaebi.",
27                  imageRes =
    R.drawable.omniscient_reader,
28                  url =
    "https://www.webtoons.com/en/action/omniscient-
    reader/list?title_no=2154"
29              ),
30              Manhwa(
31                  id = 3,
32                  title = "The Beginning After the
    End",
33                  author = "TurtleMe",
34                  description = "It follows the life
    of the late King Grey after his untimely and
    mysterious death. Reborn as Arthur Leywin, he seeks
    to correct his past mistakes in the vibrant new
    continent of Dicathen, a world of magic and
    fantastical creatures. "
35                          + "Equipped with the
    knowledge of a powerful king in his mid-thirties,
    Arthur navigates his new life as the magic-wielding
    child of two retired adventurers and gains purpose
    through each of his new experiences—something he
```

```
    lacked in his previous life. "
36                          + "When a kind dragon
    sacrifices her life to protect him, Arthur resolves
    to live a sincere, kind, and courageous life with
    those he loves. With the help of a lost elf
    princess and the Elven Kingdom of Elenoir, Arthur
    begins his long journey to find his true place in
    the world.\n\n"
37                          + "As the years pass,
    Arthur becomes more and more comfortable in this
    world, positioning himself as a young, but
    respected figure. However, deja-vu strikes as a war
    brews between Dicathen and the Vritra, a clan of
    banished deities now ruling over a faraway
    continent. "
38                          + "Arthur must rise as a
    leader, despite his fear of becoming the war-
    hardened monster he once was in his past life. "
39                          + "As the war rages on,
    Arthur discovers that he was not reborn to this
    world by chance...nor was he the only one.",
40                  imageRes = R.drawable.tbate,
41                  url =
    "https://tapas.io/series/tbate-comic/info"
42              ),
43              Manhwa(
44                  id = 4,
45                  title = "Eleceed",
46                  author = "Jeho Son",
47                  description = "Jiwoo is a kind-
    hearted young man who harnesses the lightning-quick
    reflexes of a cat to secretly make the world a
    better place - one saved little child or foster pet
    at a time. "
48                          + "Kayden is a secret agent
    on the run, who finds himself stuck in the body of
    a…um…decidedly fat old fluffy cat. "
49                          + "Together, armed with
    Jiwoo's superpowers and Kayden's uber-smarts,
    they're out to fight those forces who would let
    evil rule this world. "
50                          + "That is, if they can
    stand each other long enough to get the job done.",
51                  imageRes = R.drawable.eleceed,
52                  url =
```

```
    "https://www.webtoons.com/en/action/eleceed/list?title_no=1571"
53              ),
54              Manhwa(
55                  id = 5,
56                  title = "Killer Peter",
57                  author = "Kim Junghyun",
58                  description = "On the surface, Glory Hound is a simple human rights organization. In reality, the organization has some of the best assassins in the world, in charge of performing legendary.\n\n"
59                          + "One of their best members was simply known as Apostle Peter, and he retired in protest of the new leader, Raphael. However, resignations were not accepted, and Peter was soon ambushed. Despite his best efforts, he dies.\n\n"
60                          + "Instead of dying, though, Peter miraculously found himself back in his teenage body. He doesn't know why, but he knows one thing: he will destroy Glory Hound.",
61                  imageRes = R.drawable.killer_peter,
62                  url = "https://www.webtoons.com/en/action/killer-peter/list?title_no=5816"
63              ),
64              Manhwa(
65                  id = 6,
66                  title = "Player Who Can't Level Up",
67                  author = "GaVinGe",
68                  description = "When Kim Kigyu received his invitation to become a player (a unique-ability player, at that), he thought his struggles were over. But no matter how hard he tries, he just can’t seem to get past level 1! "
69                          + "After five years of working as a guide on the lower floors of the tower, he’s finally discovered his ability to link with “Egos” and raise his stats. "
70                          + "As his new skills unlock adventures in unexplored gates, Kigyu gets his chance to defy expectations and show the world that rank isn’t everything.",
```

```
71                  imageRes =
    R.drawable.player_cant_level_up,
72                  url =
    "https://tapas.io/episode/2414063"
73              ),
74              Manhwa(
75                  id = 7,
76                  title = "SSS-Class Revival Hunter",
77                  author = "Shinnoa",
78                  description = "After the Tower
    suddenly appeared, individuals who wished to pursue
    their personal values began to inhabit it, coming
    to be called \"hunters.\" "
79                          + "Everyone had their own
    goals, but only a chosen few were acknowledged and
    given powerful skills by the mysterious structure.
    "
80                          + "Kim Gong-Ja, a weak F-
    Class hunter without any skills, is envious of
    those who were blessed by the Tower. "
81                          + "Letting his jealousy
    overcome him one day, Gong-Ja abruptly receives a
    S-Class skill that allows him to copy a skill from
    someone else—after they kill him.\n\n"
82                          + "Sooner than he likes,
    Gong-Ja gets to test his newly acquired ability on
    the legendary hunter known as the Flame Emperor. "
83                          + "As he is dying, Gong-Ja
    learns the evil truth about the man he once admired
    the most. "
84                          + "Receiving another potent
    skill that allows him to revive and go back in time
    by 24 hours, Gong-Ja devises a plan to travel 11
    years into the past to eliminate the Flame Emperor
    and cement himself as the world's best hunter.",
85                  imageRes =
    R.drawable.sss_class_hunter,
86                  url =
    "https://www.mangaread.org/manga/sss-class-suicide-
    hunter/"
87              )
88          )
        }
89  }
90
```

Tabel 8. MainActivity.kt Modul 4

```
01  package com.example.gracemanhwa_picks
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.viewModels
7   import androidx.compose.runtime.Composable
8   import androidx.navigation.NavType
9   import androidx.navigation.compose.NavHost
10  import androidx.navigation.compose.composable
    import
    androidx.navigation.compose.rememberNavController
    import androidx.navigation.navArgument
    import
    com.example.gracemanhwa_picks.data.ManhwaRepository
    import
    com.example.gracemanhwa_picks.ui.DetailScreen
    import com.example.gracemanhwa_picks.ui.ListScreen
    import
    com.example.gracemanhwa_picks.ui.GracemanhwaPicksTh
    eme
    import
    com.example.gracemanhwa_picks.ui.ViewModel.ManhwaVi
    ewModel
    import
    com.example.gracemanhwa_picks.ui.ViewModel.ViewMode
    lFactory

    class MainActivity : ComponentActivity() {

        private val viewModel: ManhwaViewModel by
    viewModels {
            ViewModelFactory(ManhwaRepository())
        }

        override fun onCreate(savedInstanceState:
    Bundle?) {
            super.onCreate(savedInstanceState)
            setContent {
                GracemanhwaPicksTheme {
                    ManhwaApp(viewModel = viewModel)
                }
```

```
        }
    }
}

@Composable
fun ManhwaApp(viewModel: ManhwaViewModel) {
    val navController = rememberNavController()
    NavHost(navController = navController,
startDestination = "list") {
        composable("list") {
            ListScreen(navController =
navController, viewModel = viewModel)
        }

        composable(
            route = "detail/{id}",
            arguments = listOf(navArgument("id") {
type = NavType.IntType })
        ) { backStackEntry ->
            val id =
backStackEntry.arguments?.getInt("id") ?: 0
            viewModel.getManhwaById(id)?.let { item
->
                DetailScreen(item = item)
            }
        }
    }
}
```

Tabel 9. ManhwaViewModel.kt

```
1   package com.example.gracemanhwa_picks.ui.ViewModel
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import com.example.gracemanhwa_picks.data.Manhwa
6   import
7   com.example.gracemanhwa_picks.data.ManhwaRepository
8   import kotlinx.coroutines.flow.MutableStateFlow
9   import kotlinx.coroutines.flow.StateFlow
10
    class ManhwaViewModel(private val repository:
```

```
ManhwaRepository): ViewModel() {

    private val _manhwas =
MutableStateFlow<List<Manhwa>>(emptyList())
    val manhwas: StateFlow<List<Manhwa>> get() =
_manhwas

    init {
        loadManhwas()
    }

    private fun loadManhwas() {
        _manhwas.value = repository.getManhwas()
        Log.d("Manhwa ViewModel", "Manhwa data
loaded into the list.")
    }

    fun getManhwaByIdId(id: Int): Manhwa? {
        val manhwa = _manhwas.value.firstOrNull {
it.id == id}
        if (manhwa != null) {
            Log.d("Manhwa ViewModel", "Navigating
to Detail for: ${manhwa.title}")
        }
        return manhwa
    }
}
```

Tabel 10. ViewModelFactory.kt

```
01  package com.example.gracemanhwa_picks.ui.ViewModel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import
6   com.example.gracemanhwa_picks.data.ManhwaRepository
7
8   class ViewModelFactory(private val repository:
9   ManhwaRepository) : ViewModelProvider.Factory {
10      override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
        if
    (modelClass.isAssignableFrom(ManhwaViewModel::class
    .java)) {
            @Suppress("UNCHECKED_CAST")
```

```
            return ManhwaViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown
ViewModel class")
    }
}
```

Tabel 11. DetailScreen.kt

```
01  package com.example.gracemanhwa_picks.ui
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import
6   androidx.compose.foundation.rememberScrollState
7   import
8   androidx.compose.foundation.shape.RoundedCornerShap
    e
9   import androidx.compose.foundation.verticalScroll
10  import androidx.compose.material3.*
11  import androidx.compose.runtime.Composable
12  import androidx.compose.ui.Modifier
13  import androidx.compose.ui.draw.clip
14  import androidx.compose.ui.layout.ContentScale
15  import androidx.compose.ui.res.painterResource
16  import androidx.compose.ui.unit.dp
17  import com.example.gracemanhwa_picks.data.Manhwa
18
19  @Composable
20  fun DetailScreen(item: Manhwa) {
21      val scrollState = rememberScrollState()
22
23      Column(
24          modifier = Modifier
25              .fillMaxSize()
26              .verticalScroll(scrollState)
27              .padding(16.dp)
28      ) {
29          Image(
30              painter = painterResource(id =
    item.imageRes),
31              contentDescription = item.title,
32              modifier = Modifier
33                  .fillMaxWidth()
```

```
34                  .height(250.dp)
35                  .clip(RoundedCornerShape(16.dp)),
36              contentScale = ContentScale.Crop
37          )
38          Spacer(modifier = Modifier.height(16.dp))
39          Text(item.title, style =
    MaterialTheme.typography.headlineMedium)
40          Text("By ${item.author}", style =
    MaterialTheme.typography.titleMedium)
41          Spacer(modifier = Modifier.height(8.dp))
42          Divider()
43          Spacer(modifier = Modifier.height(8.dp))
44          Text(item.description, style =
    MaterialTheme.typography.bodyLarge)
45      }
46  }
```

Tabel 12. ListScreen.kt Modul 4

```
01  package com.example.gracemanhwa_picks.ui
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import
6   androidx.compose.foundation.rememberScrollState
7   import
8   androidx.compose.foundation.shape.RoundedCornerShap
    e
9   import androidx.compose.foundation.verticalScroll
10  import androidx.compose.material3.*
11  import androidx.compose.runtime.Composable
12  import androidx.compose.ui.Modifier
13  import androidx.compose.ui.draw.clip
14  import androidx.compose.ui.layout.ContentScale
15  import androidx.compose.ui.res.painterResource
16  import androidx.compose.ui.unit.dp
17  import com.example.gracemanhwa_picks.data.Manhwa
18
19  @Composable
20  fun DetailScreen(item: Manhwa) {
21      val scrollState = rememberScrollState()
```

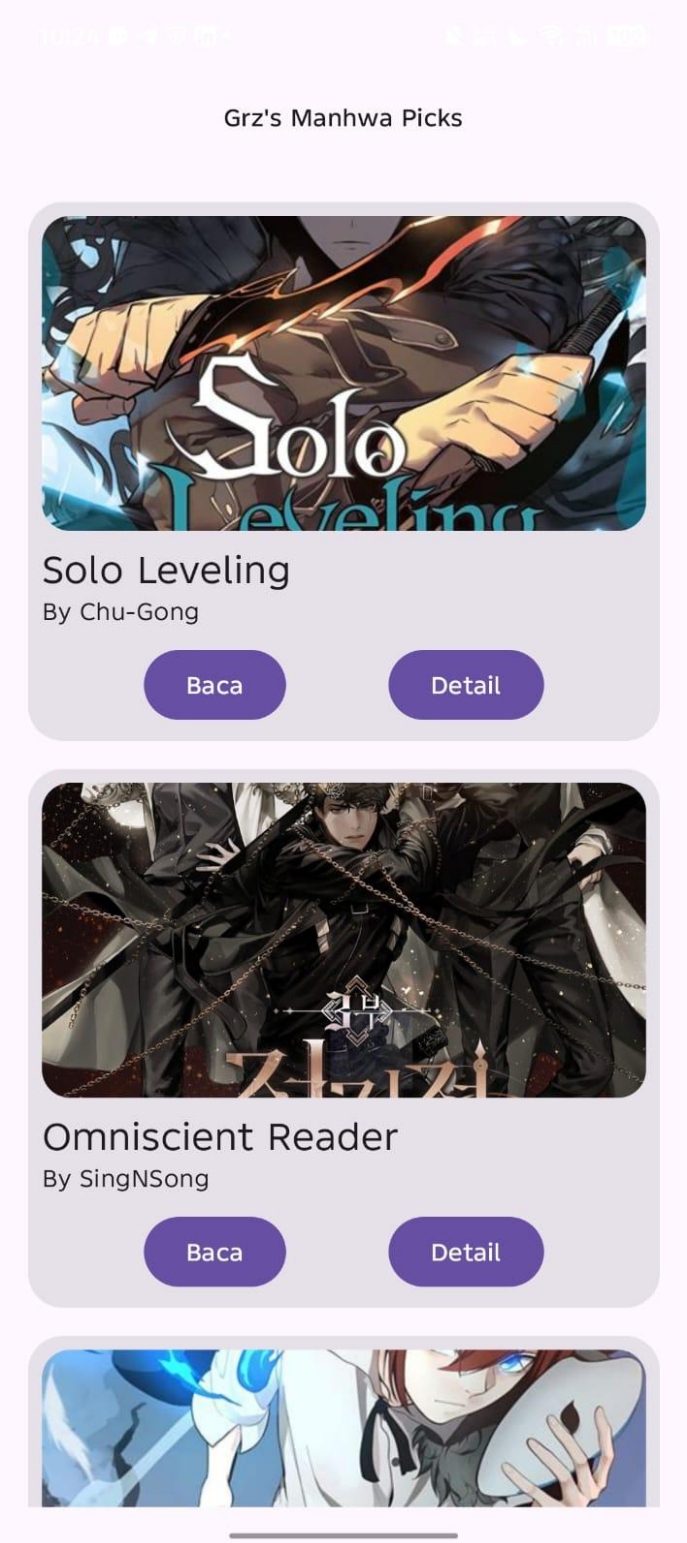
```
22      Column(
23          modifier = Modifier
24              .fillMaxSize()
25              .verticalScroll(scrollState)
26              .padding(16.dp)
27      ) {
28          Image(
29              painter = painterResource(id =
    item.imageRes),
30              contentDescription = item.title,
31              modifier = Modifier
32                  .fillMaxWidth()
33                  .height(250.dp)
34                  .clip(RoundedCornerShape(16.dp)),
35              contentScale = ContentScale.Crop
36          )
37          Spacer(modifier = Modifier.height(16.dp))
38          Text(item.title, style =
    MaterialTheme.typography.headlineMedium)
39          Text("By ${item.author}", style =
    MaterialTheme.typography.titleMedium)
40          Spacer(modifier = Modifier.height(8.dp))
41          Divider()
42          Spacer(modifier = Modifier.height(8.dp))
44          Text(item.description, style =
    MaterialTheme.typography.bodyLarge)
45      }
46  }
```

Tabel 13. Theme.kt Modul 4

```
01  package com.example.gracemanhwa_picks.ui
2
3   import androidx.compose.material3.*
4   import androidx.compose.runtime.Composable
5
6   @Composable
7   fun GracemanhwaPicksTheme(content: @Composable () -
    > Unit) {
8       MaterialTheme(
9           colorScheme = lightColorScheme(),
10          typography = Typography(),
```

```
11          content = content
12      )
13  }
```

B. Output Program



Gambar 14. Output Program Modul 4

C. Pembahasan

Manhwa.kt untuk definisi struktur data untuk objek manhwa. Pada baris pertama hingga ketiga, package dan import tidak dituliskan karena file ini tidak memerlukan import eksternal. Deklarasi data class Manhwa dimulai pada baris keempat dengan parameter seperti id, title, author, description, imageRes, dan url. Tipe data yang digunakan seluruhnya bersifat eksplisit, seperti Int untuk id dan imageRes, serta String untuk properti lainnya. Struktur ini memungkinkan penyimpanan berbagai informasi penting yang akan digunakan dalam tampilan daftar dan detail manhwa.

ManhwaRepository.kt digunakan sebagai penyedia data manhwa. Package dideklarasikan pada awal, dan import untuk resource R digunakan agar dapat mengakses gambar manhwa dari drawable. Fungsi utama di dalam file ini adalah getManhwas() yang berada pada baris keenam. Fungsi ini mengembalikan list yang terdiri dari beberapa objek Manhwa. Masing-masing objek Manhwa diinisialisasi dengan data seperti judul, penulis, deskripsi panjang, resource gambar, dan URL. Setiap manhwa memiliki deskripsi unik yang menjelaskan latar belakang cerita dan karakter utama, serta tujuan atau konflik dalam kisah tersebut. Data yang tersedia sangat lengkap, mendukung kebutuhan aplikasi untuk menampilkan informasi manhwa secara mendetail.

MainActivity.kt memulai aktivitas utama aplikasi Android. Package serta import berbagai komponen Compose dan ViewModel dideklarasikan di awal. Komponen ComponentActivity diturunkan ke class MainActivity, dan pada baris kesebelas dilakukan inisialisasi viewModel menggunakan delegate by viewModels, disertai factory ViewModelFactory yang menerima instance ManhwaRepository. Dalam fungsi onCreate, method setContent dipanggil untuk mengatur konten tampilan dengan tema GracemanhwaPicksTheme. Fungsi ManhwaApp digunakan sebagai root composable. Di dalam fungsi ManhwaApp, controller navigasi diciptakan menggunakan rememberNavController. Route "list" diatur untuk menampilkan ListScreen, sementara route "detail/{id}" akan mengambil argumen id dan menampilkan DetailScreen berdasarkan item yang sesuai dari ViewModel. Pendekatan ini mendukung arsitektur single-activity dengan navigasi antar composable yang efisien.

ManhwaViewModel.kt berisi logika ViewModel untuk mengelola data manhwa. Package dan import mendeklarasikan penggunaan ViewModel, logging, dan StateFlow. Kelas ManhwaViewModel menerima parameter repository untuk mengakses data. Di dalamnya, variabel _manhwas bertipe MutableStateFlow digunakan untuk menyimpan data secara mutable, sedangkan variabel publik manhwas bertipe StateFlow hanya menyediakan akses baca. Proses pengambilan data dimulai

dari fungsi init yang langsung memanggil loadManhwas(). Fungsi ini mengambil data dari repository dan mengisi _manhwas dengan daftar manhwa. Logging dilakukan untuk mencatat bahwa data telah dimuat. Fungsi getManhwaById menerima parameter id dan mencari objek manhwa dengan ID tersebut. Jika ditemukan, logging mencatat navigasi ke halaman detail berdasarkan judul manhwa yang dipilih.

ViewModelFactory.kt bertugas menyediakan mekanisme pembuatan ViewModel dengan parameter. Package dan import mendefinisikan penggunaan ViewModel dan ViewModelProvider. Kelas ViewModelFactory menerima parameter repository. Implementasi fungsi create digunakan untuk menghasilkan objek ManhwaViewModel jika modelClass yang diminta sesuai. Jika tidak cocok, maka dilempar pengecualian, memastikan ViewModel dapat menerima parameter eksternal dengan cara yang sesuai standar Android.

DetailScreen.kt merupakan composable untuk menampilkan halaman detail manhwa. Pada bagian awal, dilakukan import terhadap komponen layout seperti Column, Image, Text, dan Spacer, serta fungsi pendukung seperti painterResource dan ContentScale. Fungsi DetailScreen menerima parameter item bertipe Manhwa. Scroll state disiapkan agar tampilan dapat digulir secara vertikal. Layout utama menggunakan Column dengan modifier untuk ukuran penuh, scrollable, dan padding. Di dalamnya, gambar manhwa ditampilkan menggunakan Image dengan bentuk rounded dan skala cropping agar memenuhi tampilan horizontal. Selanjutnya, ditampilkan judul manhwa dengan gaya teks headline, disusul nama penulis dengan gaya teks title. Setelah itu, deskripsi panjang manhwa ditampilkan dengan pemisah berupa Divider dan jarak antar elemen dengan Spacer.

ListScreen.kt seharusnya berisi tampilan daftar manhwa, namun isi dari file yang ada adalah salinan DetailScreen.kt, menunjukkan kemungkinan kesalahan duplikasi konten saat penulisan laporan. Karena strukturnya identik, penjelasannya pun sama seperti pada DetailScreen, dan seharusnya diganti dengan kode yang menampilkan daftar manhwa dalam bentuk list item yang dapat ditekan untuk berpindah ke detail.

Theme.kt berfungsi untuk mengatur tema aplikasi. Package dan import mengarah ke penggunaan Material3 dan composable function. Fungsi GracemanhwaPicksTheme menerima lambda composable sebagai parameter. Di dalamnya, MaterialTheme digunakan dengan skema warna terang dan tipografi default. Fungsi ini membungkus semua konten aplikasi agar tetap konsisten secara tampilan dan nuansa visual.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

Jawaban Soal 2

Dalam arsitektur aplikasi Android, Application class merupakan komponen inti yang pertama kali diinisialisasi oleh sistem saat aplikasi dijalankan. Kelas ini hanya dibuat satu kali dan tetap aktif selama siklus hidup aplikasi berlangsung, sehingga sangat cocok digunakan untuk mengelola inisialisasi global. Salah satu fungsi utamanya adalah melakukan konfigurasi awal terhadap komponen atau pustaka yang digunakan di seluruh aplikasi, seperti Firebase, Retrofit, Room, atau library logging. Selain itu, Application class juga dapat dimanfaatkan untuk menyimpan state atau objek yang bersifat global, seperti repository, container dependency injection, atau konfigurasi tertentu yang diperlukan oleh berbagai aktivitas dan komponen UI.

Dengan membuat subclass dari Application dan meng-override fungsi onCreate(), proses inisialisasi yang penting dapat dilakukan sebelum aktivitas atau fragment pertama ditampilkan. Contohnya, repository atau service bisa diinisialisasi satu kali di sini, lalu disuntikkan ke dalam ViewModel melalui ViewModelFactory. Application juga bisa digunakan untuk memantau perilaku aplikasi secara keseluruhan, seperti mencatat log aktivitas, mendeteksi crash, atau mengelola analitik pengguna. Agar class ini dikenali oleh sistem Android, nama subclass Application perlu ditulis di atribut android:name pada file AndroidManifest.xml.

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

Tabel 14. ManhwaDao.kt Modul 5

```
01  package
2   com.example.gracemanhwa_picks.data.local.dao
3
4   import androidx.room.Dao
5   import androidx.room.Insert
6   import androidx.room.OnConflictStrategy
7   import androidx.room.Query
8   import androidx.room.Update
9   import
10  com.example.gracemanhwa_picks.data.local.entity.Man
    hwaEntity
11  import kotlinx.coroutines.flow.Flow
12
13  @Dao
14  interface ManhwaDao {
15      @Query("SELECT * FROM manhwas ORDER BY title
    ASC")
16      fun getAllManhwas(): Flow<List<ManhwaEntity>>
17
18      @Query("SELECT * FROM manhwas WHERE id = :id")
19      fun getManhwaById(id: Int): Flow<ManhwaEntity?>
```

```
20      @Insert(onConflict =
    OnConflictStrategy.REPLACE)
21      suspend fun insertAll(manhwas:
    List<ManhwaEntity>)
22
23      @Update
24      suspend fun updateManhwa(manhwa: ManhwaEntity)
25
26      @Query("DELETE FROM manhwas")
27      suspend fun deleteAll()
28  }
```

Tabel 15. ManhwaEntity.kt Modul 5

```
01          package
    com.example.gracemanhwa_picks.data.local.entity
2
3   import androidx.room.Entity
4   import androidx.room.PrimaryKey
5
6   @Entity(tableName = "manhwas")
7   data class ManhwaEntity(
8       @PrimaryKey
9       val id: Int,
10      val title: String,
11      val author: String,
12      val description: String,
13      val imageUrl: String,
14      val url: String,
15      val isFavorite: Boolean = false
16  )
```

Tabel 16. ManhwaDatabase.kt Modul 5

```
01  package
    com.example.gracemanhwa_picks.data.local.entity
2
3   import androidx.room.Entity
4   import androidx.room.PrimaryKey
5
6   @Entity(tableName = "manhwas")
7   data class ManhwaEntity(
8       @PrimaryKey
```

```
9       val id: Int,
10      val title: String,
11      val author: String,
12      val description: String,
13      val imageUrl: String,
14      val url: String,
15      val isFavorite: Boolean = false
16  )
```

Tabel 17. ManhwaDto.kt Modul 5

```
01  package com.example.gracemanhwa_picks.data.model

2   import kotlinx.serialization.SerialName
3   import kotlinx.serialization.Serializable
4
5   @Serializable
6   data class ManhwaDto(
7       @SerialName("id")
8       val id: Int,
9
10      @SerialName("title")
11      val title: String,
12
13      @SerialName("author")
14      val author: String,
15
16      @SerialName("description")
17      val description: String,
18
19      @SerialName("imageUrl")
20      val imageUrl: String,
21
22      @SerialName("url")
23      val url: String
24  )
```

Tabel 18. ManhwaApiService.kt Modul 5

```
01  package com.example.gracemanhwa_picks.data.model
2
3   import kotlinx.serialization.SerialName
4   import kotlinx.serialization.Serializable
```

```
5
6   @Serializable
7   data class ManhwaDto(
8       @SerialName("id")
9       val id: Int,
10
11      @SerialName("title")
12      val title: String,
13
14      @SerialName("author")
15      val author: String,
16
17      @SerialName("description")
18      val description: String,
19
20      @SerialName("imageUrl")
21      val imageUrl: String,
22
23      @SerialName("url")
24      val url: String
25  )
```

Tabel 19.RetrofitInstance.kt Modul 5

```
01  package com.example.gracemanhwa_picks.data.remote
02
3   import
    com.example.gracemanhwa_picks.data.remote.api.ManhwaApiS
    ervice
4   import
    com.jakewharton.retrofit2.converter.kotlinx.serializatio
    n.asConverterFactory
5   import kotlinx.serialization.json.Json
6   import okhttp3.MediaType.Companion.toMediaType
7   import okhttp3.OkHttpClient
8   import okhttp3.logging.HttpLoggingInterceptor
9   import retrofit2.Retrofit
10
11  object RetrofitInstance {
12
13      private const val BASE_URL =
14  "https://manhwagrzzz.free.beeceptor.com"
15
16      private val json = Json {
17          ignoreUnknownKeys = true
18      }
```

```
19
20      private val logging = HttpLoggingInterceptor().apply
21  {
22          level = HttpLoggingInterceptor.Level.BODY
23      }
24
25      private val client = OkHttpClient.Builder()
26          .addInterceptor(logging)
27          .build()
28
29      val api: ManhwaApiService by lazy {
30          Retrofit.Builder()
31              .baseUrl(BASE_URL)
32              .client(client)
33
34  .addConverterFactory(json.asConverterFactory("applicatio
    n/json".toMediaType()))
35              .build()
36              .create(ManhwaApiService::class.java)
37      }
38  }
39
```

Tabel 20.ManhwaRepository.kt

```
01  package
2   com.example.gracemanhwa_picks.data.repository
3
4   import
    com.example.gracemanhwa_picks.data.local.dao.Manhwa
    Dao
5   import
    com.example.gracemanhwa_picks.data.local.entity.Man
    hwaEntity
6   import
    com.example.gracemanhwa_picks.data.remote.api.Manhw
    aApiService
7   import kotlinx.coroutines.flow.Flow
8   import kotlinx.coroutines.flow.firstOrNull
9   import java.io.IOException
10  import android.util.Log
11
12  class ManhwaRepository(
13      private val apiService: ManhwaApiService,
```

```
14      private val manhwaDao: ManhwaDao
15  ) {
16
17      fun getAllManhwas(): Flow<List<ManhwaEntity>> =
    manhwaDao.getAllManhwas()
18
19      fun getManhwaById(id: Int): Flow<ManhwaEntity?>
    = manhwaDao.getManhwaById(id)
20
21      suspend fun refreshManhwas() {
22          try {
23              val remoteManhwas =
    apiService.getAllManhwas()
24              val favoriteManhwas =
    manhwaDao.getAllManhwas().firstOrNull()?.filter {
    it.isFavorite }?.map { it.id }
25                      ?: emptyList()
26
27              val manhwaEntities = remoteManhwas.map
28  { dto ->
29                  ManhwaEntity(
30                      id = dto.id,
31                      title = dto.title,
32                      author = dto.author,
33                      description = dto.description,
34                      imageUrl = dto.imageUrl,
35                      url = dto.url,
36                  )
37              }
38
39              manhwaDao.insertAll(manhwaEntities)
40
41          } catch (e: IOException) {
42              Log.e("ManhwaRepository", "Gagal
    refresh manhwas (IO): ", e)
43              e.printStackTrace()
44          } catch (e: Exception) {
45              Log.e("ManhwaRepository", "Gagal
    refresh manhwas (Exception): ", e)
46              e.printStackTrace()
47          }
48      }
49      suspend fun updateFavoriteStatus(manhwa:
    ManhwaEntity, isFavorite: Boolean) {
50
```

```
51  manhwaDao.updateManhwa(manhwa.copy(isFavorite =
    isFavorite))
52      }
53  }
```

Tabel 21. Injection.kt Modul 5

```
01  package com.example.gracemanhwa_picks.di
2
3   import android.content.Context
4   import
    com.example.gracemanhwa_picks.data.local.ManhwaData
    base
5   import
    com.example.gracemanhwa_picks.data.remote.RetrofitI
    nstance
6   import
    com.example.gracemanhwa_picks.data.repository.Manhw
    aRepository
7
8   object Injection {
9       fun provideRepository(context: Context):
10  ManhwaRepository {
11          val database =
    ManhwaDatabase.getDatabase(context)
12          val apiService = RetrofitInstance.api
13          return ManhwaRepository(apiService,
    database.manhwaDao())
14      }
15  }
```

Tabel 22. ManhwaCard.kt Modul 5

```
001 package
-   com.example.gracemanhwa_picks.presentation.compone
    nts
2
3   import android.content.Intent
4   import android.net.Uri
5   import androidx.compose.foundation.clickable
6   import androidx.compose.foundation.layout.*
7   import
    androidx.compose.foundation.shape.RoundedCornerSha
```

```
    pe
8   import androidx.compose.material.icons.Icons
9   import
    androidx.compose.material.icons.filled.Favorite
10  import
    androidx.compose.material.icons.outlined.FavoriteB
    order
11  import androidx.compose.material3.*
12  import androidx.compose.runtime.Composable
13  import androidx.compose.ui.Alignment
14  import androidx.compose.ui.Modifier
15  import androidx.compose.ui.graphics.Color
16  import androidx.compose.ui.layout.ContentScale
17  import androidx.compose.ui.platform.LocalContext
18  import androidx.compose.ui.text.font.FontWeight
19  import androidx.compose.ui.text.style.TextOverflow
20  import androidx.compose.ui.unit.dp
21  import androidx.compose.ui.unit.sp
22  import coil.compose.AsyncImage
23  import
    com.example.gracemanhwa_picks.data.local.entity.Ma
    nhwaEntity
24
25  @Composable
26  fun ManhwaCard(
27      manhwa: ManhwaEntity,
28      onFavoriteClick: () -> Unit,
29      onDetailClick: () -> Unit,
30      modifier: Modifier = Modifier
31  ) {
32      val context = LocalContext.current
33
34      Card(
35          modifier = modifier
36              .fillMaxWidth()
37              .clickable { onDetailClick() },
38          shape = RoundedCornerShape(16.dp),
39          elevation =
    CardDefaults.cardElevation(defaultElevation =
40  4.dp)
41      ) {
            Box(modifier =
42  Modifier.height(IntrinsicSize.Min)) {
43              Row(
                    verticalAlignment =
```

```
44  Alignment.CenterVertically
45              ) {
46                  AsyncImage(
47                      model = manhwa.imageUrl,
                        contentDescription =
48  manhwa.title,
                        contentScale =
49  ContentScale.Crop,
                        modifier =
50  Modifier.width(120.dp).fillMaxHeight()
51                  )
52                  Column(
                        modifier =
53  Modifier.padding(16.dp)
54                  ) {
55                      Text(
56                          text = manhwa.title,
                            style =
57  MaterialTheme.typography.titleMedium,
                            fontWeight =
58  FontWeight.Bold,
59                          maxLines = 2,
                            overflow =
60  TextOverflow.Ellipsis
61                      )
                        Spacer(modifier =
62  Modifier.height(4.dp))
63                      Text(
                            text = "By
64  ${manhwa.author}",
                            style =
65  MaterialTheme.typography.bodySmall,
66                          color = Color.Gray
67                      )
                        Spacer(modifier =
68  Modifier.height(8.dp))
69                      Row(
                            modifier =
70  Modifier.fillMaxWidth(),
                            horizontalArrangement =
71  Arrangement.spacedBy(8.dp)
72                      ) {
73                          OutlinedButton(
                                onClick =
74  onDetailClick,
```

```
                                modifier =
75  Modifier.weight(1f)
                            ) { Text("Detail",
76  fontSize = 12.sp) }
77
78                          Button(
79                              onClick = {
                                    val intent =
80  Intent(Intent.ACTION_VIEW, Uri.parse(manhwa.url))
81  context.startActivity(intent)
82                              },
83                              modifier =
    Modifier.weight(1f)
84                          ) { Text("Baca", fontSize
85  = 12.sp) }
86                      }
87                  }
88              }
89              IconButton(
90                  onClick = onFavoriteClick,
91                  modifier = Modifier
92                      .align(Alignment.TopEnd)
93                      .padding(8.dp)
94              ) {
95                  Icon(
                        imageVector = if
    (manhwa.isFavorite) Icons.Filled.Favorite else
96  Icons.Outlined.FavoriteBorder,
                        contentDescription = "Toggle
97  Favorite",
98                      tint = if (manhwa.isFavorite)
99  Color.Red else Color.White
100                 )
101             }
            }
        }
    }
```

Tabel 23. Navigation.kt Modul 5

```
01  package
-   com.example.gracemanhwa_picks.presentation.navigatio
    n
```

```
2
3   import androidx.compose.runtime.Composable
4   import
    androidx.lifecycle.viewmodel.compose.viewModel
5   import androidx.navigation.NavHostController
6   import androidx.navigation.NavType
7   import androidx.navigation.compose.NavHost
8   import androidx.navigation.compose.composable
9   import androidx.navigation.navArgument
10  import
    com.example.gracemanhwa_picks.presentation.screen.Ma
    nhwaDetailScreen
11  import
    com.example.gracemanhwa_picks.presentation.screen.Ma
    nhwaListScreen
12  import
    com.example.gracemanhwa_picks.presentation.viewmodel
    .ManhwaViewModel
13  import
    com.example.gracemanhwa_picks.presentation.viewmodel
    .ManhwaViewModelFactory

14  @Composable
15  fun AppNavigation(
16      navController: NavHostController,
17      factory: ManhwaViewModelFactory
18  ) {
19      val viewModel: ManhwaViewModel =
20  viewModel(factory = factory)
21
22      NavHost(navController = navController,
    startDestination = "list") {
23          composable("list") {
24              ManhwaListScreen(
25                  viewModel = viewModel,
26                  onNavigateToDetail = { manhwaId ->
27           navController.navigate("detail/$manhwaId")
28                  }
29              )
30          }
31          composable(
32              route = "detail/{manhwaId}",
33              arguments =
    listOf(navArgument("manhwaId") { type =
    NavType.IntType })
```

```
        ) { backStackEntry ->
34          val manhwaId =
backStackEntry.arguments?.getInt("manhwaId") ?: 0
35          ManhwaDetailScreen(
36              manhwaId = manhwaId,
37              viewModel = viewModel,
38              onNavigateBack = {
39                  navController.popBackStack()
40              }
41          )
42      }
43  }
44  }
```

Tabel 24. ManhwaDetailScreen.kt Modul 5

```
01  package
-   com.example.gracemanhwa_picks.presentation.screen

2   import androidx.compose.foundation.layout.*
3   import
    androidx.compose.foundation.rememberScrollState
4   import
    androidx.compose.foundation.shape.RoundedCornerShape
5   import androidx.compose.foundation.verticalScroll
6   import androidx.compose.material.icons.Icons
7   import
    androidx.compose.material.icons.automirrored.filled.
    ArrowBack
8   import androidx.compose.material3.*
9   import androidx.compose.runtime.Composable
10  import androidx.compose.runtime.LaunchedEffect
11  import androidx.compose.runtime.collectAsState
12  import androidx.compose.runtime.getValue
13  import androidx.compose.ui.Alignment
14  import androidx.compose.ui.Modifier
15  import androidx.compose.ui.draw.clip
16  import androidx.compose.ui.layout.ContentScale
17  import androidx.compose.ui.unit.dp
18  import coil.compose.AsyncImage
19  import
    com.example.gracemanhwa_picks.presentation.viewmodel
    .ManhwaViewModel
20
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun ManhwaDetailScreen(
    manhwaId: Int,
    viewModel: ManhwaViewModel,
    onNavigateBack: () -> Unit
) {
    LaunchedEffect(manhwaId) {
        viewModel.getManhwaById(manhwaId)
    }

    val manhwa by
viewModel.selectedManhwa.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text(manhwa?.title ?:
"Detail Manhwa") },
                navigationIcon = {
                    IconButton(onClick =
onNavigateBack) {
Icon(Icons.AutoMirrored.Filled.ArrowBack,
contentDescription = "Kembali")
                    }
                }
            )
        }
    ) { innerPadding ->
        manhwa?.let { item ->
            Column(
                modifier = Modifier
                    .padding(innerPadding)
                    .fillMaxSize()
              .verticalScroll(rememberScrollState())
                    .padding(16.dp)
            ) {
                AsyncImage(
                    model = item.imageUrl,
                    contentDescription = item.title,
                    contentScale =
ContentScale.Crop,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(300.dp)
```

```
62
63  .clip(RoundedCornerShape(16.dp))
64                  )
65                  Spacer(modifier =
66  Modifier.height(16.dp))
67                  Text(item.title, style =
    MaterialTheme.typography.headlineMedium)
68                  Text("By ${item.author}", style =
    MaterialTheme.typography.titleMedium)
69                  Spacer(modifier =
    Modifier.height(8.dp))
70                  Divider()
                    Spacer(modifier =
71  Modifier.height(8.dp))
72                  Text(item.description, style =
    MaterialTheme.typography.bodyLarge)
73              }
            } ?: run {
74              Box(
75                  modifier = Modifier
76                      .padding(innerPadding)
77                      .fillMaxSize(),
78                  contentAlignment = Alignment.Center
79              ) {
80                  CircularProgressIndicator()
81              }
82          }
83      }
84  }
```

Tabel 25. ManhwaViewModel.kt Modul 5

```
01  package
-   com.example.gracemanhwa_picks.presentation.viewmode
    l
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import
    com.example.gracemanhwa_picks.data.local.entity.Man
    hwaEntity
6   import
    com.example.gracemanhwa_picks.data.repository.Manhw
    aRepository
7   import kotlinx.coroutines.flow.MutableStateFlow
```

```
8  import kotlinx.coroutines.flow.StateFlow
9  import kotlinx.coroutines.flow.asStateFlow
10 import kotlinx.coroutines.launch
11
12 class ManhwaViewModel(private val repository:
13 ManhwaRepository) : ViewModel() {
14
15     private val _manhwas =
   MutableStateFlow<List<ManhwaEntity>>(emptyList())
16     val manhwas: StateFlow<List<ManhwaEntity>> =
   _manhwas.asStateFlow()
17
18     private val _selectedManhwa =
   MutableStateFlow<ManhwaEntity?>(null)
19     val selectedManhwa: StateFlow<ManhwaEntity?> =
   _selectedManhwa.asStateFlow()
20
21     private val _isLoading =
   MutableStateFlow(false)
22     val isLoading: StateFlow<Boolean> =
   _isLoading.asStateFlow()
23
24     init {
25         loadManhwas(forceRefresh = true)
26     }
27
28     private fun loadManhwas(forceRefresh: Boolean)
   {
29         viewModelScope.launch {
30             if (forceRefresh) {
31                 _isLoading.value = true
32                 repository.refreshManhwas()
33             }
34             repository.getAllManhwas().collect {
   manhwaList ->
35                 _manhwas.value = manhwaList
36                 _isLoading.value = false
37             }
38         }
39     }
40
41     fun getManhwaById(id: Int) {
42         viewModelScope.launch {
43             repository.getManhwaById(id).collect {
44                 _selectedManhwa.value = it
```

```
45              }
46          }
47      }
48
49      fun toggleFavorite(manhwa: ManhwaEntity) {
50          viewModelScope.launch {
51              repository.updateFavoriteStatus(manhwa,
    !manhwa.isFavorite)
52          }
53      }
54  }
```

Tabel 26. ManhwaViewModelFactory.kt Modul 5

```
01  package
-   com.example.gracemanhwa_picks.presentation.viewmode
    l
2
3   import android.content.Context
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6   import com.example.gracemanhwa_picks.di.Injection
7
8   class ManhwaViewModelFactory(private val context:
    Context) : ViewModelProvider.Factory {
        override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
9           if
    (modelClass.isAssignableFrom(ManhwaViewModel::class
    .java)) {
10              @Suppress("UNCHECKED_CAST")
11              return
    ManhwaViewModel(Injection.provideRepository(context
    )) as T
12          }
13          throw IllegalArgumentException("Unknown
14  ViewModel class")
15      }
16  }
```

Tabel 27. Theme.kt Modul 5

```
01- package com.example.gracemanhwa_picks.ui.theme

2   import android.app.Activity
3   import android.os.Build
4   import
5   androidx.compose.foundation.isSystemInDarkTheme
6   import androidx.compose.material3.MaterialTheme
7   import androidx.compose.material3.darkColorScheme
8   import
    androidx.compose.material3.dynamicDarkColorScheme
9   import
    androidx.compose.material3.dynamicLightColorScheme
10  import androidx.compose.material3.lightColorScheme
11  import androidx.compose.runtime.Composable
12  import androidx.compose.ui.platform.LocalContext
13
14  private val DarkColorScheme = darkColorScheme(
15      primary = Purple80,
16      secondary = PurpleGrey80,
17      tertiary = Pink80
18  )
19
20  private val LightColorScheme = lightColorScheme(
21      primary = Purple40,
22      secondary = PurpleGrey40,
23      tertiary = Pink40
24  )
25
26  @Composable
27  fun GraceManhwa_picksTheme(
28      darkTheme: Boolean = isSystemInDarkTheme(),
29      dynamicColor: Boolean = true,
30      content: @Composable () -> Unit
31  ) {
32      val colorScheme = when {
33          dynamicColor && Build.VERSION.SDK_INT >=
    Build.VERSION_CODES.S -> {
34              val context = LocalContext.current
35              if (darkTheme)
    dynamicDarkColorScheme(context) else
    dynamicLightColorScheme(context)
37          }
38
39          darkTheme -> DarkColorScheme
```

```
40          else -> LightColorScheme
41      }
42
43      MaterialTheme(
44          colorScheme = colorScheme,
45          typography = Typography,
46          content = content
47      )
48  }
```

Tabel 28. ConnectivityObserver.kt

```
1   package com.example.modul5.util
2
3   import kotlinx.coroutines.flow.Flow
4
5   interface ConnectivityObserver {
6       fun observe(): Flow<Status>
7
8       enum class Status {
9           Available, Unavailable, Losing, Lost
10      }
11  }
```

Tabel 29. MainActivity.kt Modul 5

```
01  package com.example.gracemanhwa_picks
-
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import
    androidx.compose.foundation.layout.fillMaxSize
7   import androidx.compose.material3.MaterialTheme
8   import androidx.compose.material3.Surface
9   import androidx.compose.ui.Modifier
10  import
    androidx.navigation.compose.rememberNavController
11  import
    com.example.gracemanhwa_picks.presentation.navigatio
    n.AppNavigation
12  import
```

```
    com.example.gracemanhwa_picks.presentation.viewmodel
    .ManhwaViewModelFactory
13  import
    com.example.gracemanhwa_picks.ui.theme.GraceManhwa_p
    icksTheme
14
15  class MainActivity : ComponentActivity() {
16      override fun onCreate(savedInstanceState:
    Bundle?) {
17          super.onCreate(savedInstanceState)
18          setContent {
19              GraceManhwa_picksTheme {
20                  Surface(
21                      modifier =
    Modifier.fillMaxSize(),
22                      color =
    MaterialTheme.colorScheme.background
23                  ) {
24                      val navController =
    rememberNavController()
25                      val factory =
    ManhwaViewModelFactory(this)
26
27                      AppNavigation(navController =
    navController, factory = factory)
28                  }
29              }
30          }
31      }
32  }
```

B. Output Produk

C. Pembahasan

ManhwaDao.kt berisi interface ManhwaDao yang mendefinisikan fungsi-fungsi akses data lokal menggunakan Room. Fungsi-fungsi seperti getAllManhwas(), getManhwaById(), insertAll(), updateManhwa(), dan deleteAll() digunakan untuk mengambil, menyimpan, dan memperbarui data manhwa secara reaktif melalui Flow. ManhwaEntity.kt berisi data class ManhwaEntity sebagai representasi dari entitas

dalam database lokal Room. Setiap entitas menyimpan data manhwa seperti id, title, author, description, imageUrl, url, dan isFavorite. Struktur ini memudahkan penyimpanan dan pengelolaan data secara offline.

ManhwaDto.kt berisi data class ManhwaDto yang berfungsi untuk mendeskripsikan struktur data dari API eksternal. Dengan anotasi @Serializable dan @SerialName, KotlinX Serialization digunakan untuk memetakan data JSON ke dalam objek Kotlin. RetrofitInstance.kt berisi konfigurasi singleton Retrofit untuk komunikasi HTTP. Tabel ini menyiapkan JSON parser yang toleran terhadap field tak dikenal (ignoreUnknownKeys = true), logging interceptor, dan pembuatan objek Retrofit dengan ManhwaApiService.

ManhwaRepository.kt berisi layer repository yang menghubungkan data dari API dan database lokal. Repository ini mengimplementasikan caching, sinkronisasi data, dan error handling. Ia juga menyediakan fungsi refreshManhwas() dan updateFavoriteStatus() untuk sinkronisasi data dan pengelolaan status favorit. Injection.kt berisi class Injection sebagai penyedia dependensi ManhwaRepository secara manual. Fungsi provideRepository() mengambil instance database lokal dan Retrofit untuk membentuk repository.

ManhwaCard.kt sebagai komponen UI berbasis Jetpack Compose yang menampilkan kartu informasi manhwa. Tabel ini memuat elemen visual seperti gambar, judul, nama penulis, tombol detail, tombol baca, dan ikon favorit yang dapat diklik. Navigation.kt berisi konfigurasi NavHost menggunakan Jetpack Compose Navigation. Aplikasi memiliki dua layar: daftar manhwa (list) dan detail manhwa (detail/{manhwaId}), dengan pengelolaan argument dan ViewModel menggunakan factory.

ManhwaDetailScreen.kt menampilkan layar detail dari suatu manhwa tertentu. Menggunakan AsyncImage, Text, dan Scaffold, layar ini menampilkan informasi lengkap manhwa dan memungkinkan kembali ke layar sebelumnya dengan ikon kembali. ManhwaViewModel.kt berisi ViewModel yang mengatur aliran data antara repository dan UI. Menyediakan StateFlow untuk semua manhwa dan manhwa yang dipilih, serta fungsi untuk memuat data awal, mengambil manhwa berdasarkan ID, dan mengubah status favorit.

ManhwaViewModelFactory.kt menampilkan factory class yang digunakan untuk menghasilkan instance ManhwaViewModel dengan dependensi yang tepat dari Injection. Hal ini diperlukan untuk integrasi dengan sistem ViewModel Compose. Theme.kt berisi konfigurasi tema aplikasi. Digunakan untuk menerapkan skema warna terang dan gelap secara otomatis atau dinamis, sesuai pengaturan sistem dan API level perangkat.

ConnectivityObserver.kt menampilkan interface yang memungkinkan observasi status konektivitas jaringan secara reaktif. Interface ini berisi fungsi observe() dan enum Status untuk memantau kondisi seperti Available, Lost, dan lainnya. MainActivity.kt dalamnya dipanggil GraceManhwa_picksTheme, diset Surface, dan digunakan AppNavigation untuk mengelola navigasi antar layar menggunakan NavController.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

natnutnot/PrakMobile at master + https://github.com/natnutnot/Mobile