LAPORAN AKHIR PRAKTIKUM
PEMROGAMAN MOBILE

Oleh:

Damarjati Suryo Laksono NIM. 2310817210014

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 : Viewmodel And Debugging
Modul 5 : Connect To The Internet

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

Nama Praktikan : Damarjati Suryo Laksono
NIM : 2310817210014

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

DAFTAR ISI

DAFTAR GAMBAR

Modul 1 : Android Basic With Kotlin

Modul 2 : Android Layout

Modul 3 : Build A Scrollable List

Modul 4 : Viewmodel And Debugging

Modul 5 : Connect To The Internet

DAFTAR TABEL

Modul 1 : Android Basic With Kotlin

Modul 2 : Android Layout

Modul 3 : Build A Scrollable List

Modul 4 : Viewmodel And Debugging

Modul 5 : Function dan Database

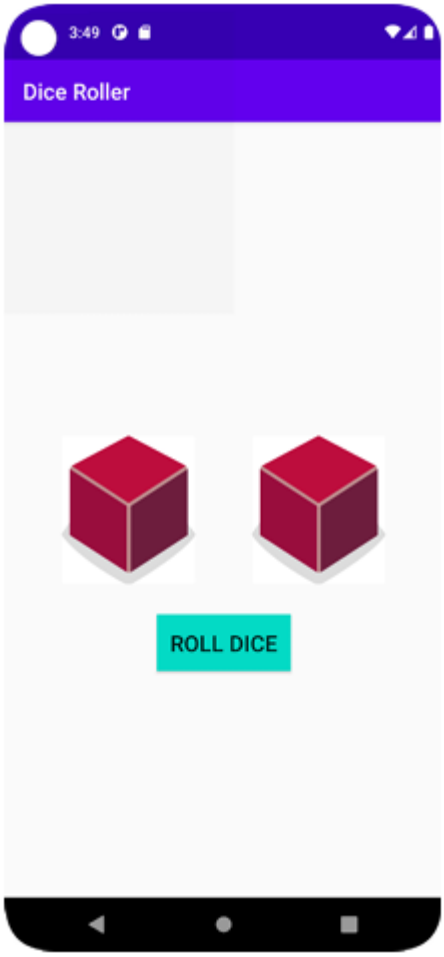
MODUL 1 : ANDROID BASIC WITH KOTLIN

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.

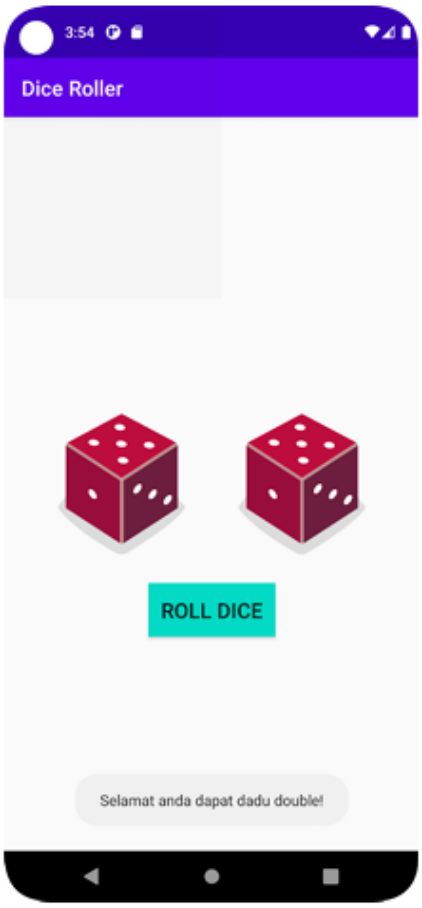


Gambar 1. 1. Tampilan Awal Aplikasi

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.

3. 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.

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



Gambar 1. 2. Tampilan Roll Dadu Double

A. Source Code

1. MainActivity.kt

```
1   package com.example.diceroller
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.viewModels
6   import androidx.appcompat.app.AppCompatActivity
7   import androidx.lifecycle.Observer
8   import
    com.example.diceroller.databinding.ActivityMainBindin
    g
9
10  class MainActivity : AppCompatActivity() {
11
12      private lateinit var binding :ActivityMainBinding
        private val viewModel : DiceViewModel by
13  viewModels()

14      override fun onCreate(savedInstanceState:
15  Bundle?) {
            super.onCreate(savedInstanceState)
16          binding =
17  ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)
18
19          binding.button.setOnClickListener {
20              viewModel.rollDice()
21          }
22          viewModel.dice1.observe(this,Observer { value
23  ->

    binding.imageView.setImageResource(getDiceImage(value
24  ))
            })
```

```
25          viewModel.dice2.observe(this,Observer { value
26  ->

    binding.imageView2.setImageResource(getDiceImage(valu
27  e))
            })
28
    viewModel.isDouble.observe(this,Observer{ isDouble ->
29              if (isDouble){
                    Toast.makeText(this,"Selamat anda
30  dapat dadu double", Toast.LENGTH_SHORT).show()
31              }else{
                    Toast.makeText(this,"Anda belum
    beruntung", Toast.LENGTH_SHORT).show()
32              }
33          })
        }
34
35      private fun getDiceImage(value: Int):Int{
36          return when(value){
37              1 ->R.drawable.dice_1
38              2 ->R.drawable.dice_2
39              3 ->R.drawable.dice_3
40              4 ->R.drawable.dice_4
41              5 ->R.drawable.dice_5
42              else -> R.drawable.dice_6
43          }
44      }
45  }
46
47
48
```

Tabel 1. 1. Source Code MainActivity.kt

2. DiceViewModel.kt

```
1   package com.example.diceroller
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
5   import androidx.lifecycle.ViewModel
6   import kotlin.random.Random
7
8   class DiceViewModel : ViewModel() {
9       private val _dice1 = MutableLiveData(1)
10      val dice1 : LiveData<Int>
11          get() = _dice1
12
13      private val _dice2 = MutableLiveData(1)
14      val dice2 : LiveData<Int>
15          get() = _dice2
16
17      private val _isDouble = MutableLiveData(false)
        val isDouble : LiveData<Boolean>
18          get() = _isDouble
19
20      fun rollDice() {
21          val roll1 = Random.nextInt(1,7)
22          val roll2 = Random.nextInt(1,7)
23          _dice1.value = roll1
24          _dice2.value = roll2
25          _isDouble.value = roll2 == roll1
26      }
27  }
28
```

Tabel 1. 2. Source Code DiceViewModel.kt

3. activity_main.xml

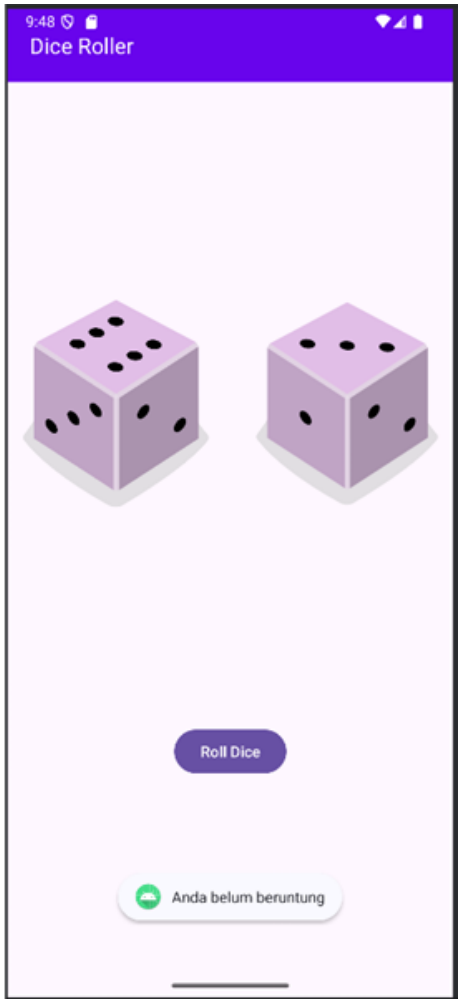
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/and
    roid"
        xmlns:app="http://schemas.android.com/apk/res-
    auto"
3       xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/main"
        android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       tools:context=".MainActivity">
6
7       <TextView
8           android:id="@+id/headerTitle"
9           android:layout_width="0dp"
10          android:layout_height="wrap_content"
11          android:background="#6804ec"
12          android:padding="20dp"
13          android:text="@string/app_name"
14          android:textColor="#FFFFFF"
15          android:textSize="20sp"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintHorizontal_bias="0.0"
18          app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
19
        <LinearLayout
20          android:id="@+id/diceContainer"
            android:layout_width="wrap_content"
21          android:layout_height="wrap_content"
            android:orientation="horizontal"
22          android:gravity="center"
23
24  app:layout_constraintTop_toBottomOf="@+id/headerTitle
```

```
25  "
26
27  app:layout_constraintBottom_toTopOf="@+id/button"
28          app:layout_constraintStart_toStartOf="parent"
29          app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintVertical_bias="0.5">
30
            <ImageView
                android:id="@+id/imageView"
31              android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/dice_0" />
32
            <ImageView
33              android:id="@+id/imageView2"
                android:layout_width="wrap_content"
34              android:layout_height="wrap_content"
35              android:src="@drawable/dice_0"
36              android:layout_marginStart="16dp" />
37      </LinearLayout>
38
39      <Button
40          android:id="@+id/button"
41          android:layout_width="wrap_content"
42          android:layout_height="wrap_content"
43          android:text="@string/button_1"
44
45  app:layout_constraintTop_toBottomOf="@+id/diceContain
46  er"
47
48  app:layout_constraintBottom_toBottomOf="parent"
49          app:layout_constraintStart_toStartOf="parent"
50          app:layout_constraintEnd_toEndOf="parent"
51          app:layout_constraintVertical_bias="0.0"
52          android:layout_marginBottom="32dp" />
```

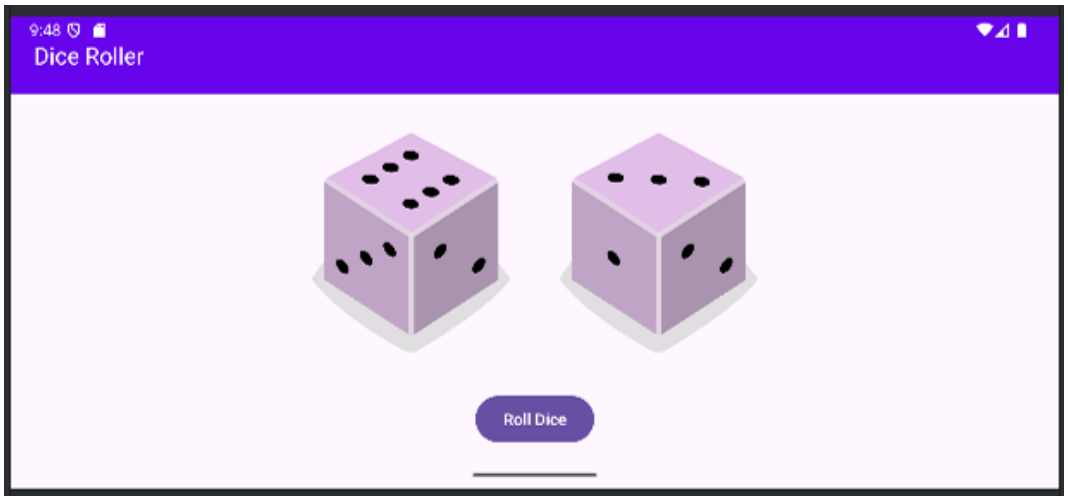
```
53
54  </androidx.constraintlayout.widget.ConstraintLayout>
55
56
57
58
59
60
61
62
```

Tabel 1. 3. Source Code activity_main.xml

B. Output Program



Gambar 1. 3. Screenshot Vertikal Aplikasi



Gambar 1. 4. Screenshoot Horizontal Aplikasi

C. Pembahasan

1. MainActivity.kt:

Pada line [1] – [8], terdapat package dan library – library yang akan di gunakan.

Pada line [10], terdapat main activity yang merupakan wadah dari kode yang akan menjadi wadah untuk kode lainnya.

Pada line [12] – [13] terdapat inisasi awal dari view binding dan view model.

Pada line [15] – [18] terdapat insiasi dari view binding.

Pada line [20] – [22] terdapat syntax `binding.button.setOnClickListener`, pada syntax ini, button di set ke listener atau Ketika di klik akan melakukan aksi yang di tentukan, Dimana aksi disini merupakan `rollDice()`, namun pada syntax button tadi terdapat binding.button yang secara otomatis mengarah ke button yang terdapat pada `main_activity.xml` tanpa harus menulis sintaks yang panjang. Kemudian pada `rollDice()` juga terdapat `viewModel.rollDice()`, sintaks ini mengarah ke fungsi `rollDice()` pada file `DiceViewModel.kt`, fungsi ini bertujuan untuk mengacak angka dadu dari 1-6.

Pada line [23] – [28] terdapat syntax dengan kerangka seperti `viewModel.dice1.observe(this,Observer{value-> binding.imageView.setImageResource(getDiceImage(value))`, pertama `viewModel.dice1` mengarah pada variable dice1 yang terdapat di DiceViewModel.kt, kemudian terdapat syntax `.observe(this,Observer{value->` syntax ini bertujuan untuk menyatakan pada program bahwa program harus mengamati value pada kasus ini `dice1`, jika value `dice1` pada `DiceViewModel.kt` berubah maka akan terjadi perubahan pada image, selanjutnya terdapat syntax `binding.imageView.setImageResource(getDiceImage(value))`, Dimana syntax ini bertujuan untuk merubah image sesuai dengan jumlah mata dadu yang terdapat pada `dice1`, pada syntax ini terdapat fungsi `getDiceImage`, Dimana fungsi ini berguna untuk mengambil gambar sesuai dengan mata dadu, setelah itu melakukan

`setImageResource` yang betujuan untuk merubah gambar dari `imageView`.

Pada line [29] – [36] terdapat syntax `getDiceImage()` yang berguna untuk mengecek apakah dadu double atau tidak, pertama program akan mengambil data `isDouble` dari `DiceViewModel.kt`, kemudian program akan melakukan observasi terdapat data `isDouble` tersebut, apabila ternyata data adalah `true`, maka program akan memberi selamat, dan jika ternyata adalah `false` maka program akan memberi celotehan, cara kerja pengiriman pesna tersebut adalah dengan melakukan print notifkasi menggunakan `Toast`, dan pada akhir terdapat `.show()` untuk menunjukan pesan.

Pada line [38] – [48] terdapat fungsi `getDiceImage()` untuk menentukan gambar mana yang di gunakan berdasar hasil roll dadu.

2. DiceViewModel.kt

Pada line [1] – [6], terdapat package dan library – library yang akan di gunakan.

Pada line [8], terdapat `class DiceViewModel : ViewModel()` yang merupakan wadah dari kode yang akan menjadi wadah untuk kode lainnya, dan juga menentukan bahwa class ini merupakan `Viewmodel()` dari projek ini.

Pada line [9] – [19], terdapat syntax `private val _dice1 = MutableLiveData(1)`, pada syntax ini terdapat `private val _dice1` yang menyebutkan bahwa `value _dice1` hanya bisa di baca di class ini, dan penggunaan pada `_dice1` penamaan ini menandakan bahwa variable ini adalah variable local saja, setelah itu terdapat `MutableLiveData(1)`, yang berfungsi agar data bisa di rubah namun hanya pada lingkup local, dan (1) merupakan place holder data itu sendiri, kemudian terdapat `val dice1 : LiveData get() = _dice1`, Dimana `val dice1 : LiveData` berguna untuk melakukan deklarasi bahwa `dice1` bersifat unmutable dan hanya bisa di observe atau di lihat

aja , kemudian terdapat `get() = _dice1`, syntax ini bertujuan untuk mengisi data `dice1`, menjadi data dari `_dice1` yang akan berubah di fungsi dibawah,dan untuk variable lainnya, cara kerjanya juga sama.

Pada line [21] – [28], terdapat fungsi fun `rollDice()`, fungsi ini bertujuan untuk melakukan rolling pada dadu menggunakan syntax `val roll1 = Random.nextInt(1,7)`, dan juga menentukan apakah dadu double atau tidak dan di simpan ke dalam Boolean menggunakan `_isDouble.value = roll2 == roll1`.

3. activity_main.xml

Pada line [1], terdapat deklarasi `<?xml version="1.0" encoding="utf 8"?>`, yang merupakan standar deklarasi file XML agar dikenali dengan benar oleh sistem Android.

Pada line [2] – [8], terdapat tag Utama `<androidx.constraintlayout.widget.ConstraintLayout >`, yaitu ini adalah root layout yang digunakan untuk menyusun elemen-elemen UI secara fleksibel dengan constraint dan juga layout ini punya id, `layout_width`, `layout_height`, dan `tools:context` untuk menunjuk ke `MainActivity`.

Pada line [10] – [20], terdapat komponen `TextView` dengan id headerTitle, yang berfungsi sebagai header aplikasi.

Pada line [22] – [38], terdapat `LinearLayout` dengan id diceContainer yang berfungsi sebagai wadah untuk dua gambar dadu.

Pada line [24] – [33], ada dua `ImageView` yang merepresentasikan dua buah dadu.

Pada line [40] – [48], terdapat sebuah `Button` dengan id button yang digunakan untuk me-roll dadu.

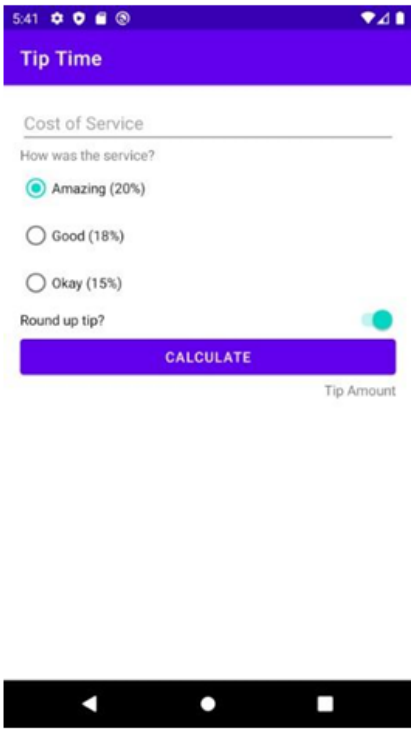
MODUL 2 : ANDROID LAYOUT

SOAL 1

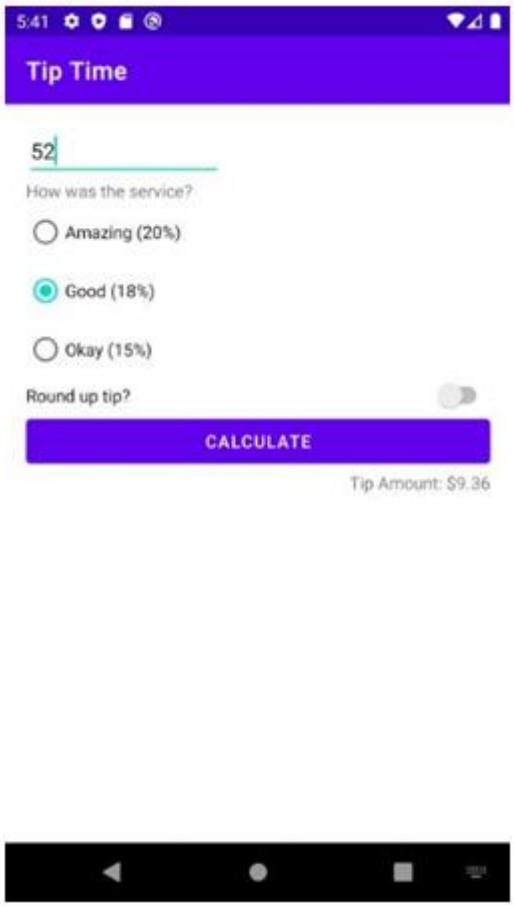
Soal Praktikum:

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 2. 1. Tampilan Awal Aplikasi



Gambar 2. 2. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1. MainActivity.kt

```
1  package com.example.tiptime
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.activity.viewModels
6  import androidx.appcompat.app.AppCompatActivity
7  import androidx.appcompat.app.AppCompatDelegate
8  import
   androidx.core.splashscreen.SplashScreen.Companion.ins
   tallSplashScreen
9  import androidx.lifecycle.Observer
```

```
10  import
    com.example.tiptime.databinding.ActivityMainBinding

11  class MainActivity : AppCompatActivity() {
12
13      private lateinit var binding :
14  ActivityMainBinding
        private val viewModel: TipViewModel by
15  viewModels()

16      override fun onCreate(savedInstanceState:
17  Bundle?) {
            super.onCreate(savedInstanceState)
18          installSplashScreen()
19
20  AppCompatDelegate.setDefaultNightMode(AppCompatDelega
21  te.MODE_NIGHT_NO)

22          binding =
23  ActivityMainBinding.inflate(layoutInflater)
            setContentView(binding.root)
24
25          binding.filledButton.setOnClickListener{
26              val input =
27  binding.EditText.text?.toString()?.trim()
                val isRounded =
28  binding.tipround.isChecked
                val tipPercentage = when
29  (binding.radiobuttonGroup.checkedRadioButtonId){
                    R.id.option1 -> 0.20
30                  R.id.option2 -> 0.18
31                  R.id.option3 -> 0.15
32                  else -> null
33              }
34              viewModel.calculateTip(input,
```

```
35  tipPercentage, isRounded)
        }
36
37  viewModel.tipResult.observe(this,Observer { result ->
38              binding.resultText.text = result
        })
39  viewModel.errorMessage.observe(this,Observer{ message
40  ->
41              message?.let {
     Toast.makeText(this,it,Toast.LENGTH_SHORT).show()
42                  viewModel.errorMessageHandled()
                }
43          })
        }
44  }
45
46
47
48
```

Tabel 2. 1. Source Code MainActivity.kt

2. TipViewModel.kt

```
1   package com.example.tiptime
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
5   import androidx.lifecycle.ViewModel
6   import kotlin.math.ceil
7
8   class TipViewModel: ViewModel() {
9       private val _tipResult =
    MutableLiveData<String>()
10      val tipResult: LiveData<String> = _tipResult
11
```

```
12      private val _errorMessage =
    MutableLiveData<String?>()
13      val errorMessage: LiveData<String?> =
    _errorMessage
14
        fun calculateTip(ammountInput:
15  String?,tipPercentage: Double?, isRounded: Boolean){
            val ammount = ammountInput?.toDoubleOrNull()
16          if (ammount == null || tipPercentage ==
    null){
17              _errorMessage.value = "Input Harus Berupa
    Angka"
18              return
            }
19          if(ammount < 0){
20              _errorMessage.value = "Angka tidak boleh
21  negatif"
22              return
            }
23
24          val tip = ammount * tipPercentage
25          val total = if (isRounded) ceil(tip) else tip
26          _tipResult.value = "Tip Ammount $$total"
27      }

28      fun errorMessageHandled() {
29          _errorMessage.value = null
30      }
31
32  }
33
34
35
```

Tabel 2. 2. Source Code TipViewModel.kt

3. TipViewModel.kt

```
1   <ScrollView
    xmlns:android="http://schemas.android.com/apk/res/and
    roid"
2   xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       android:background="@color/white"
6       tools:context=".MainActivity">
7   <androidx.constraintlayout.widget.ConstraintLayout
8           android:id="@+id/main"
            android:layout_width="match_parent"
9           android:layout_height="match_parent"
10          android:background="@color/white"
11          tools:context=".MainActivity">
12          <TextView
13              android:id="@+id/headerTitle"
14              android:layout_width="0dp"
15              android:layout_height="wrap_content"
16              android:background="#6804ec"
17              android:padding="20dp"
18              android:text="@string/app_name"
19              android:textColor="#FFFFFF"
20              android:textSize="20sp"
21  app:layout_constraintEnd_toEndOf="parent"
22  app:layout_constraintHorizontal_bias="0.0"
23  app:layout_constraintStart_toStartOf="parent"
24  app:layout_constraintTop_toTopOf="parent" />
25
26  <com.google.android.material.textfield.TextInputLayou
27  t
28              android:id="@+id/textField"
                android:layout_width="match_parent"
29              android:layout_height="wrap_content"
```

```
30              android:paddingTop="16dp"
31              android:paddingLeft="8dp"
32              android:paddingRight="8dp"
33              android:hint="@string/label"
34              app:helperTextEnabled="true"
35              app:helperText="@string/helper_text"
36              app:endIconMode="clear_text"
37              app:errorEnabled="true"
38  app:layout_constraintTop_toBottomOf="@id/headerTitle"
39  app:layout_constraintStart_toStartOf="parent"
40  app:layout_constraintEnd_toEndOf="parent">

41  <com.google.android.material.textfield.TextInputEditT
42  ext
43              android:id="@+id/EditText"
44              android:layout_width="match_parent"
                android:layout_height="wrap_content"
45              />
46  </com.google.android.material.textfield.TextInputLayo
47  ut>
48
49          <RadioGroup
                android:id="@+id/radiobutton_group"
50  android:checkedButton="@+id/enabled_selected"
51              android:layout_width="0dp"
52              android:layout_height="wrap_content"
53  app:layout_constraintWidth_percent="0.95"
54  app:layout_constraintTop_toBottomOf="@+id/textField"
55  app:layout_constraintStart_toStartOf="parent"
56  app:layout_constraintEnd_toEndOf="parent">
57
                <RadioButton
58                  android:id="@+id/option1"
59                  android:checked="true"
60                android:layout_width="match_parent"
```

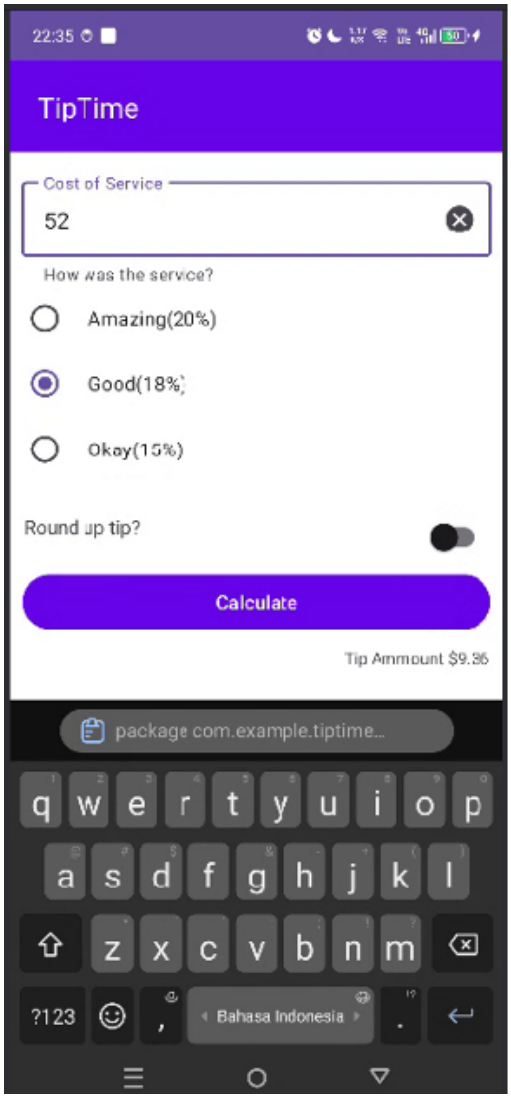
```
61                  android:layout_height="match_parent"
62                  android:enabled="true"
63  android:paddingStart="@dimen/padding_start"
    android:paddingEnd="@dimen/padding_start"
64  android:text="@string/radiobutton_text1"
                    android:tag="0.20"/>
65              <RadioButton
66                  android:id="@+id/option2"
67
68   android:layout_width="match_parent"
69                  android:layout_height="match_parent"
70                  android:checked="false"
71                  android:enabled="true"
72  android:paddingStart="@dimen/padding_start"
    android:paddingEnd="@dimen/padding_start"
73  android:text="@string/radiobutton_text2"
                    android:tag="0.18"/>
74              <RadioButton
75                  android:id="@+id/option3"
76                android:layout_width="match_parent"
77  android:layout_height="match_parent"
78                  android:checked="false"
79                  android:enabled="true"
80  android:paddingStart="@dimen/padding_start"
81  android:paddingEnd="@dimen/padding_start"
82  android:text="@string/radiobutton_text3"
                    android:tag="0.15"/>
83
84          </RadioGroup>
85
86          <TextView
87              android:id="@+id/tipLabel"
88              android:layout_width="0dp"
89              android:layout_height="wrap_content"
90              android:text="@string/label_1"
```

```
91              android:gravity="start"
92              android:textSize="14dp"
93              android:layout_marginTop="1dp"
94              android:layout_marginEnd="8dp"
95  app:layout_constraintStart_toStartOf="parent"
96
97  app:layout_constraintTop_toBottomOf="@id/radiobutton_
98  group"
99  app:layout_constraintBottom_toBottomOf="@id/tiproun
100 app:layout_constraintEnd_toStartOf="@id/tiproun
101             android:paddingStart="10dp"/>
102
103 <com.google.android.material.switchmaterial.SwitchMat
    erial
104             android:id="@+id/tiproun
                android:layout_width="wrap_content"
105             android:layout_height="wrap_content"
                android:layout_marginTop="16dp"
106             android:paddingRight="10dp"
    app:layout_constraintBottom_toTopOf="@id/filledButton
107 "           app:layout_constraintEnd_toEndOf="parent"
108 app:layout_constraintTop_toBottomOf="@id/radiobutton_
109 group" />

110         <Button
111             android:backgroundTint="#6804ec"
112             android:id="@+id/filledButton"
113             android:layout_width="0dp"
114             android:layout_height="wrap_content"
115             android:text="@string/button_text"
    app:layout_constraintWidth_percent="0.95"
116 app:layout_constraintStart_toStartOf="parent"
117 app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tiproun
118 />
```

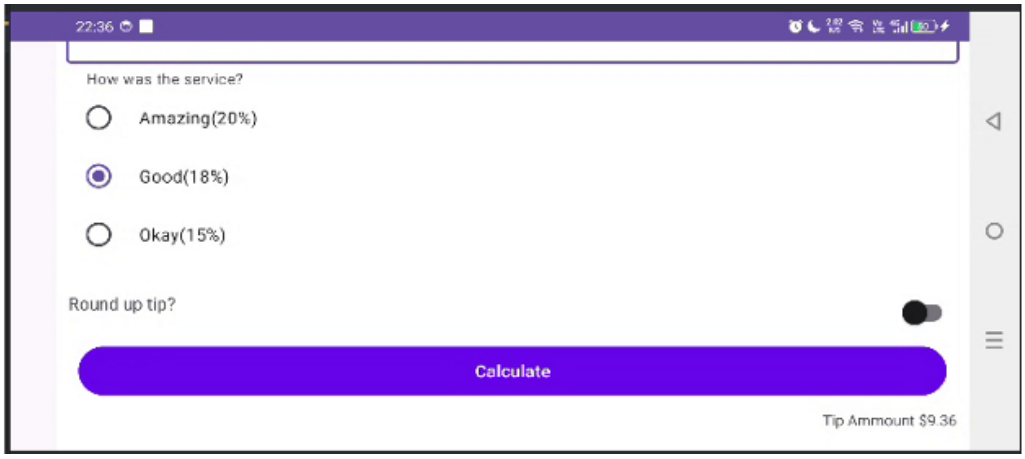
```
119
120         <TextView
121             android:id="@+id/result_text"
122             android:layout_width="150dp"
123             android:layout_height="33dp"
124             android:textSize="12dp"
125             android:layout_marginTop="8dp"
126             android:paddingEnd="10dp"
127             android:gravity="right"
128             android:text="@string/result_text"
    app:layout_constraintTop_toBottomOf="@+id/filledButto
129 n"
130 app:layout_constraintEnd_toEndOf="parent"
131             android:textFontWeight="600"/>
132  </androidx.constraintlayout.widget.ConstraintLayout>
133 </ScrollView>
14
135
136
137
138
139


140
141
142
143


144
```

Tabel 2. 3. Source Code activity_main.xml

B. Output Program



Gambar 2. 3. Screenshot Vertikal Aplikasi



Gambar 2. 4. Screenshoot Horizontal Aplikasi

C. Pembahasan

1. MainActivity.kt:

Pada line [1] – [10], terdapat package dan library – library yang akan di gunakan Pada line [12], terdapat main activity yang merupakan wadah dari kode yang akan Pada line [14] – [15] terdapat inisasi awal dari view binding dan view model

Pada line [19] terdapat inisiasi splashscreen untuk android dengan sdk 31+

Pada line [21] terdapat fungsi untuk memaksa pengguna nightmode agar tetap menggunakan theme light

Pada line [23] – [24] terdapat insiasi dari view binding

Pada line [26] – [36] terdapat syntax `binding.button.setOnClickListener`, pada syntax ini, button di set ke listener atau Ketika di klik akan melakukan aksi yang di tentukan, Dimana aksi disini akan menjalankan fungsi `calculateTip` pada `viewModel` dengan parameter `input`, `tipPercentage`, dan `isRounded`, dan juga pada line ini juga terdapat variable seperti `val input` di gunakan untuk menyimpan secara sementara input dari user,kemudian juga terdapat variable `isRounded` yang di gunakan untuk mengecek apakah user melakukan klik pada round option pada interface, dan juga terdapat `tipPercentage` yang di gunakan untuk menyimpan pilihan tip yang akan di berikan oleh user.

Pada line [38] – [40] terdapat fungsi seperti berikut

viewModel.tipResult.observe(this,*Observer* { result ->

```
    binding.resultText.text = result
```

Fungsi diatas ditujukan untuk melakukan checking apakah terdapat data di dalam variable `tipResult` yang terdapat pada `viewModel`, jika terdapat maka xml yang berkode `resultText` akan di rubah

sesuai dengan `result` yang tersedia pada `viewModel`.

Pada line [41] – [46] terdapat fungsi sebagai berikut

```
viewModel.errorMessage.observe(this,Observer{
    message -> message?.let{

Toast.makeText(this,it,Toast.LENGTH_SHORT).show(
        ) viewModel.errorMessageHandled()
    }
})
```

Selanjutnya pada fungsi diatas digunakan untuk melakukan toast ketika `errorMessage` pada `viewModel` terisi, dan kemudian jika `Toast` sudah di tampilkan, maka `errorMessage` akan di bersihkan dengan fungsi `errorMessageHandled()` pada `view model`

A. TipViewModel.kt

Pada line [1] – [6], terdapat package dan library – library yang akan di gunakan Pada line [8], terdapat `class TipViewModel: ViewModel()` yang merupakan wadah dari kode yang akan menjadi wadah untuk kode lainnya, dan juga menentukan bahwa class ini merupakan `Viewmodel()` dari projek ini.

Pada line [9] – [13], terdapat syntax `private val _tipResult = MutableLiveData<String>()`, pada syntax ini terdapat `private val _tipResult` yang menyebutkan bahwa value `_tipResult` hanya bisa di baca di class ini, dan penggunaan `_` pada `_tipResult` penamaan ini menunjukan secara eksplisit bahwa variable ini adalah variable local, setelah itu terdapat `MutableLiveData<String>()`, yang berfungsi agar data bisa di

rubah namun hanya pada lingkup local. kemudian terdapat `val tipResult: LiveData<String> = _tipResult`, Dimana `val val tipResult: LiveData<String>` berguna untuk melakukan deklarasi bahwa `tipResult` bersifat unmutable dan hanya bisa di observe atau di lihat aja, begitu juga dengan syntax sejenisnya.

Pada line [15] – [29], terdapat fungsi `fun calculateTip()`, dengan parameter `ammountInput`, `tipPercentage`, dan `isRounded`, dan data yang dimasukan bisa bersifat `null` atau `kosong` untuk data `ammountInput` dan `tipPercentage.` fungsi ini bekerja dengan cara pertama pada line [16] `ammountTip` didefinisikan menjadi entah `double` atau `null`, kemudian pada baris [17] – [24] terdapat validasi apakah `ammount` merupakan angka atau bukan dan apakah `amount` merupakan bilangan negative atau tidak, jika terindikasi benar maka `errorMessage` akan di isi dengan text yang sudah disediakan, selanjutnya pada baris [26] –[29] terdapat logika sederhana untuk melakukan kalkulasi tip itu sendiri.

Pada line [31] – [33] terdapat fungsi `errorMessageHandled()`, yang jika di panggil maka akan mengganti value dari `errorMessage` menjadi `null` atau kosong.

B. activity_main.xml

Pada line [1], terdapat scroll view yang digunakan untuk melakukan scrolling pada aplikasi

Pada line [8] – [143] terdapat `layout constraint` yang digunakan sebagai layout utama pada modul ini, dan kemudian pada baris [14]-[26] terdapat `textView` yang digunakan sebagai `header`, selanjutnya pada baris [28] –[49] terdapat `textfield.TextInputLayout` yang digunakan sebagai wadah untuk `TextInputEditText` yang berguna sebagai input field pada

layout ini, kemudian pada baris [51] –[92] terdapat `RadioGroup` dan `RadioButton`, dan pada baris

[94] –[117] terdapat `textView` dan custom switch dari google, dan pada baris [119] –[141] terdapat `button` dan `textView`.

D. Additional feature

- viewBinding
- Toast when error is present
- Splashscreen animation
- Logo

MODUL 3: BUILD A SCROLLABLE LIST

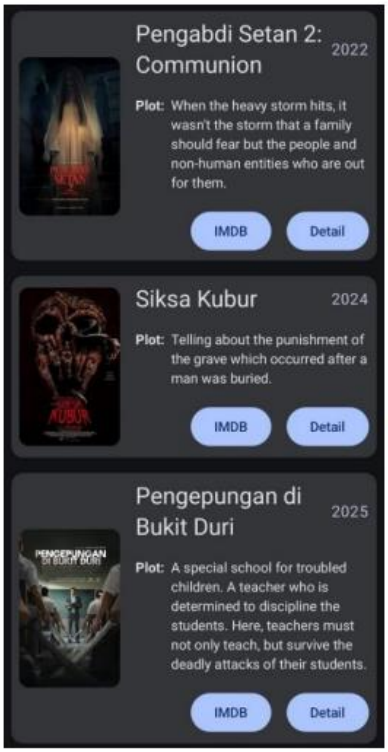
SOAL 1

Soal Praktikum:

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

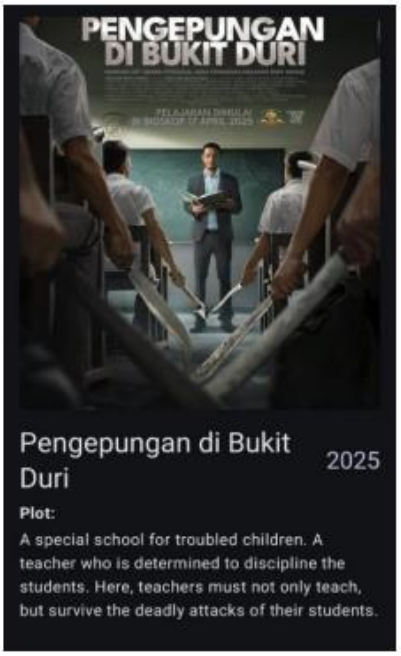
2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

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



Gambar 3. 1. Contoh UI List

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



Gambar 3. 2. Gambar Ui Detail

Simpan dengan nama file: PRAK301.PHP

A. Source Code

1. MainActivity.kt

```
1  package com.example.myrecylerview
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.appcompat.app.AppCompatActivity
6  import androidx.appcompat.app.AppCompatDelegate
7  import
   androidx.core.splashscreen.SplashScreen.Companion.inst
8  allSplashScreen
9  import
10 com.example.myrecylerview.databinding.ActivityMainBind
11 ing
12
13 class MainActivity : AppCompatActivity() {
14
```

```
15      private lateinit var binding: ActivityMainBinding
16
17      override fun onCreate(savedInstanceState: Bundle?)
18  {
19
20  AppCompatDelegate.setDefaultNightMode(AppCompatDelegat
21  e.MODE_NIGHT_NO)
            installSplashScreen()
22
23          super.onCreate(savedInstanceState)
24          binding =
25  ActivityMainBinding.inflate(layoutInflater)
26          setContentView(binding.root)
27
28          binding.btnMenu.setOnClickListener {
29              Toast.makeText(this, "This Button is used
30  for Decoration", Toast.LENGTH_SHORT).show()
            }
        }
    }
```

Tabel 3. 1. Source Code MainActivity.kt

2. DetailFragment.kt

```
1   package com.example.myrecylerview
2
3   import android.content.Intent
4   import android.os.Bundle
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import androidx.fragment.app.Fragment
9   import androidx.navigation.fragment.findNavController
    import
10  com.example.myrecylerview.databinding.FragmentDetailBi
```

```
nding

class DetailFragment : Fragment() {

    private var _binding: FragmentDetailBinding? =
null
    private val binding get() = _binding!!

    override fun onCreateView(
        inflater: LayoutInflater, container:
ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding =
FragmentDetailBinding.inflate(inflater, container,
false)
        return binding.root
    }

    override fun onViewCreated(view: View,
savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val args =
DetailFragmentArgs.fromBundle(requireArguments())
        val photo = args.extraPhoto
        val link = args.extraLink
        val detail = args.extraDetail

        binding.detailImage.setImageResource(photo)
        binding.detailDescription.text = detail

        binding.btnBack.setOnClickListener {
            findNavController().navigateUp()
        }
```

```
45
46          binding.btnShare.setOnClickListener {
47              val shareText = buildString {
48                  append("Check this
49  out!\n\n$detail\n\nMore info: $link")
50              }
51
52              val shareIntent =
53  Intent(Intent.ACTION_SEND).apply {
54                  type = "text/plain"
55                  putExtra(Intent.EXTRA_TEXT, shareText)
56              }
57
58
59  startActivity(Intent.createChooser(shareIntent, "Share
60  via"))
61          }
62      }
63
      override fun onDestroyView() {
          super.onDestroyView()
          _binding = null
      }
  }
```

Tabel 3. 2. Source Code DetailFragment.kt

3. MyAdapter.kt

```
1  package com.example.myrecylerview
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.RecyclerView
6  import
```

```
7  com.example.myrecylerview.databinding.ItemLayoutBindin
8  g
9
10 class MyAdapter(
11     private val listCharacter: ArrayList<MyData>,
12     private val onYTClick: (String) -> Unit,
13     private val onDetailClick: (String, Int, String) -
14 > Unit
15 ) : RecyclerView.Adapter<MyAdapter.ListViewHolder>() {
16
17     class ListViewHolder(val binding:
18 ItemLayoutBinding) :
19 RecyclerView.ViewHolder(binding.root)
20
21     override fun onCreateViewHolder(parent: ViewGroup,
22 viewType: Int): ListViewHolder {
23         val binding =
24 ItemLayoutBinding.inflate(LayoutInflater.from(parent.c
25 ontext), parent, false)
26         return ListViewHolder(binding)
27     }
28
29     override fun getItemCount(): Int =
30 listCharacter.size
31
32     override fun onBindViewHolder(holder:
33 ListViewHolder, position: Int) {
34         val (name, link, photo, detail, subtext) =
35 listCharacter[position]
36         with(holder.binding) {
37             textTitle.text = name
38             textDescription.text = subtext
39             itemImage.setImageResource(photo)
40             btnWebsite.setOnClickListener
    { onYTClick(link) }
```

```
            btnDetails.setOnClickListener
{ onDetailClick(detail, photo, link) }
        }
    }
}
```

Tabel 3. 3. Source Code MyAdapter.kt

4. MyFragment.kt

```
1  package com.example.myrecylerview
2
   import android.content.Intent
3  import android.net.Uri
4  import android.os.Bundle
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import androidx.fragment.app.Fragment
9  import androidx.navigation.fragment.findNavController
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import
12 com.example.myrecylerview.databinding.FragmentMyBinding
13
14
15 class MyFragment : Fragment() {
16
17     private var _binding: FragmentMyBinding? = null
18     private val binding get() = _binding!!
19
20     private lateinit var characterAdapter: MyAdapter
21     private val list = ArrayList<MyData>()
22
23     override fun onCreateView(
```

```
        inflater: LayoutInflater, container:
ViewGroup?,
        savedInstanceState: Bundle?
    ): View {
        _binding = FragmentMyBinding.inflate(inflater,
container, false)
        list.clear()
        list.addAll(getListCharacter())
        setupRecyclerView()
        return binding.root
    }

    private fun setupRecyclerView() {
        characterAdapter = MyAdapter(
            list,
            onYTClick = { link ->
                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(link))
                startActivity(intent)
            },
            onDetailClick = { detail, photo, link ->
                val action = MyFragmentDirections
                    .actionMyFragmentToDetailFragment(p
hoto, link, detail)
                findNavController().navigate(action)
            }
        )

        binding.rvCharacter.apply {
            layoutManager =
LinearLayoutManager(requireContext())
            adapter = characterAdapter
            setHasFixedSize(true)
        }
    }
```

```
59
60      private fun getListCharacter(): ArrayList<MyData> {
61          val dataName =
62  resources.getStringArray(R.array.data_name)
63          val dataLink =
64  resources.getStringArray(R.array.data_link)
65          val dataSubtext =
66  resources.getStringArray(R.array.data_subtext)
67          val dataPhoto =
68  resources.obtainTypedArray(R.array.data_photo)
69          val dataDetail =
70  resources.getStringArray(R.array.data_detail)
71          val listCharacter = ArrayList<MyData>()
72          for (i in dataName.indices) {
73              val character = MyData(
74                  name = dataName[i],
75                  link = dataLink[i],
76                  subtext = dataSubtext[i],
77                  detail = dataDetail[i],
78                  photo = dataPhoto.getResourceId(i, -1)
79              )
80              listCharacter.add(character)
81          }
82          dataPhoto.recycle()
83          return listCharacter
84      }
85
86      override fun onDestroyView() {
87          super.onDestroyView()
88          _binding = null
89      }
    }
```

Tabel 3. 4. Source Code MyFragment.kt

5. MyData.kt

```
1  package com.example.myrecylerview
2
   import android.os.Parcelable
3  import kotlinx.parcelize.Parcelize
4
5  @Parcelize
6  data class MyData(
7      val name: String,
8      val link: String,
9      val photo: Int,
10     val detail: String,
11     val subtext: String
12
13 ):Parcelable
```

Tabel 3. 5. Source Code MyData.kt

6. activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@android:color/white"
8      tools:context=".MainActivity">
9
10     <com.google.android.material.appbar.MaterialToolbar
11         android:id="@+id/toolbar"
12         android:layout_width="match_parent"
13         android:layout_height="?attr/actionBarSize"
14         android:background="@android:color/white"
```

```
15          android:elevation="4dp"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:title="Owi Core"
20          app:titleTextColor="@android:color/black"
21
22  app:titleTextAppearance="@style/TextAppearance.MaterialComponents.Hea
23
24          <ImageView
25              android:id="@+id/btn_menu"
26              android:layout_width="24dp"
27              android:layout_height="24dp"
28              android:layout_gravity="end"
29              android:layout_marginEnd="16dp"
30              android:src="@drawable/ic_more_vert"
31              app:tint="@android:color/black" />
32
33      </com.google.android.material.appbar.MaterialToolbar>
34
35      <androidx.fragment.app.FragmentContainerView
36          android:id="@+id/nav_host_fragment"
37          android:name="androidx.navigation.fragment.NavHostFragment"
38          android:layout_width="0dp"
39          android:layout_height="0dp"
40          app:layout_constraintTop_toBottomOf="@id/toolbar"
41          app:layout_constraintBottom_toBottomOf="parent"
42          app:layout_constraintStart_toStartOf="parent"
43          app:layout_constraintEnd_toEndOf="parent"
44          app:navGraph="@navigation/nav_graph"
45          app:defaultNavHost="true" />
46
47  </androidx.constraintlayout.widget.ConstraintLayout>
48
49
```

Tabel 3. 6. Source Code activity_main.xml

7. fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.core.widget.NestedScrollView

3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:fillViewport="true">
9
10      <androidx.constraintlayout.widget.ConstraintLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:padding="16dp">
14
15          <ImageButton
16              android:id="@+id/btn_back"
17              android:layout_width="48dp"
18              android:layout_height="48dp"
19
20  android:background="?selectableItemBackgroundBorderless"
21              android:src="@drawable/ic_arrow_back"
22              app:layout_constraintStart_toStartOf="parent"
23              app:layout_constraintTop_toTopOf="parent"
24              app:tint="@android:color/black" />
25
26
27  <com.google.android.material.imageview.ShapeableImageView
28              android:id="@+id/detail_image"
29              android:layout_width="200dp"
30              android:layout_height="200dp"
```

```
31              android:scaleType="centerCrop"
32              app:layout_constraintEnd_toEndOf="parent"
33              app:layout_constraintStart_toStartOf="parent"
34              app:layout_constraintTop_toTopOf="parent"
35              app:layout_constraintVertical_bias="0"
36              android:layout_marginTop="32dp"
37              android:elevation="4dp"
38
39  android:transitionName="shared_image_container"/>
40
41          <LinearLayout
42              android:layout_width="match_parent"
43              android:layout_height="wrap_content"
44              android:orientation="vertical"
45              android:padding="24dp"
46              app:layout_constraintEnd_toEndOf="parent"
47              app:layout_constraintStart_toStartOf="parent"
48
49  app:layout_constraintTop_toBottomOf="@id/detail_image"
50              android:layout_marginTop="24dp">
51
52              <androidx.cardview.widget.CardView
53                  android:layout_width="match_parent"
54                  android:layout_height="wrap_content"
55                  app:cardCornerRadius="16dp"
56                  app:cardElevation="4dp"
57
58  app:cardBackgroundColor="@android:color/white"
59                  android:layout_marginBottom="16dp">
60
61                  <TextView
62                      android:id="@+id/detail_description"
63                      android:layout_width="match_parent"
64                      android:layout_height="wrap_content"
65                      android:textSize="16sp"
```

```
                    android:lineSpacingMultiplier="1.2"
                    android:textColor="@android:color/black"
                    android:padding="24dp"
                    tools:text="Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Sed do eiusmod tempor
incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi
ut aliquip ex ea commodo consequat."/>

            </androidx.cardview.widget.CardView>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:gravity="center_horizontal"
                android:spacing="16dp">


<com.google.android.material.button.MaterialButton
                    android:id="@+id/btn_share"

style="@style/Widget.Material3.Button.OutlinedButton"
                    android:layout_width="match_parent"
                    android:layout_height="48dp"
                    android:text="Share"
                    app:icon="@drawable/ic_share"
                    app:iconTint="@null"
                    app:cornerRadius="8dp"/>

            </LinearLayout>

        </LinearLayout>

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

```
101
102 </androidx.core.widget.NestedScrollView>
```

Tabel 3. 7. Source Code fragment_detail.xml

8. fragment_my.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MyFragment">
8
9
10     <androidx.recyclerview.widget.RecyclerView
11         android:id="@+id/rv_character"
12         android:layout_width="0dp"
13         android:layout_height="0dp"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18 </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 3. 8. Source Code fragment_my.xml

9. item_layout.xml

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

3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="2dp"
9       app:cardCornerRadius="12dp"
10      app:cardElevation="4dp"
11      app:cardUseCompatPadding="true">
12
13      <androidx.constraintlayout.widget.ConstraintLayout
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:padding="16dp">
17
18          <ImageView
19              android:src="@drawable/pakdhe_1"
20              android:id="@+id/item_image"
21              android:layout_width="64dp"
22              android:layout_height="64dp"
23              android:scaleType="centerCrop"
24              app:layout_constraintBottom_toBottomOf="parent"
25              app:layout_constraintStart_toStartOf="parent"
26              app:layout_constraintTop_toTopOf="parent"
27              android:layout_marginEnd="16dp"/>
28
29          <LinearLayout
30              android:layout_width="0dp"
31              android:layout_height="wrap_content"
32              android:orientation="vertical"
33
34  app:layout_constraintBottom_toBottomOf="@id/item_image"
35
36  app:layout_constraintEnd_toStartOf="@+id/button_group"
37
38  app:layout_constraintStart_toEndOf="@id/item_image"
39
```

```
40  app:layout_constraintTop_toTopOf="@id/item_image">
41
42              <TextView
43                  android:id="@+id/text_title"
44                  android:layout_width="match_parent"
45                  android:layout_height="wrap_content"
46                  android:layout_marginStart="8dp"
47                  android:ellipsize="end"
48                  android:maxLines="1"
49                  android:textColor="@android:color/black"
50                  android:textSize="18sp"
51                  android:textStyle="bold"
52                  tools:text="Main Title" />
53
54              <TextView
55                  android:id="@+id/text_description"
56                  android:layout_width="match_parent"
57                  android:layout_height="wrap_content"
58                  android:layout_marginStart="8dp"
59                  android:layout_marginTop="4dp"
60                  android:ellipsize="end"
61                  android:maxLines="2"
62
63  android:textColor="@android:color/darker_gray"
64                  android:textSize="14sp"
65                  tools:text="Secondary description text that
66  can span multiple lines" />
67          </LinearLayout>
68
69          <LinearLayout
70              android:id="@+id/button_group"
71              android:layout_width="wrap_content"
72              android:layout_height="wrap_content"
73              android:orientation="vertical"
74              android:gravity="end"
```

```
            android:spacing="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent">


<com.google.android.material.button.MaterialButton
                android:id="@+id/btn_details"

style="@style/Widget.Material3.Button.OutlinedButton"
                android:layout_width="wrap_content"
                android:layout_height="36dp"
                android:minWidth="64dp"
                app:icon="@drawable/ic_info_outline"
                app:iconPadding="0dp"
                app:iconGravity="textStart"
                app:cornerRadius="8dp"/>


<com.google.android.material.button.MaterialButton
                android:id="@+id/btn_website"

style="@style/Widget.Material3.Button.TonalButton"
                android:layout_width="wrap_content"
                android:layout_height="36dp"
                android:minWidth="64dp"
                app:icon="@drawable/ic_open_in_browser"
                app:iconPadding="0dp"
                app:iconGravity="textStart"
                app:cornerRadius="8dp"/>
        </LinearLayout>

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

Tabel 3. 9. Source Code item_layout.xml

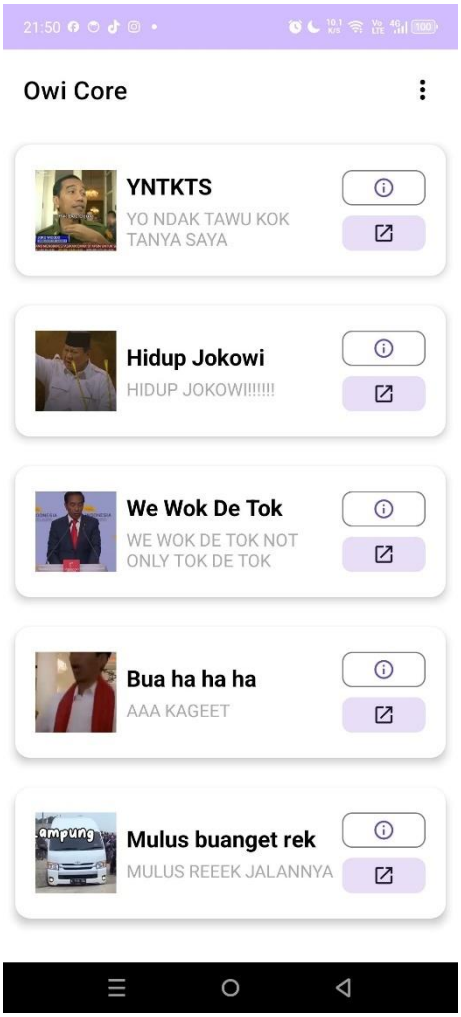
10. nav_graph.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <navigation
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      android:id="@+id/nav_graph"
6      app:startDestination="@id/myFragment">
7
8      <fragment
9          android:id="@+id/myFragment"
10         android:name="com.example.myrecylerview.MyFragment"
11         android:label="MyFragment" >
12         <action
13
14 android:id="@+id/action_myFragment_to_detailFragment"
15             app:destination="@id/detailFragment" />
16     </fragment>
17
18     <fragment
19         android:id="@+id/detailFragment"
20
21 android:name="com.example.myrecylerview.DetailFragment"
22         android:label="Detail">
23
24         <argument
25             android:name="extraPhoto"
26             app:argType="integer" />
27
28         <argument
29             android:name="extraLink"
30             app:argType="string" />
31
32         <argument
33             android:name="extraDetail"
34             app:argType="string" />
```

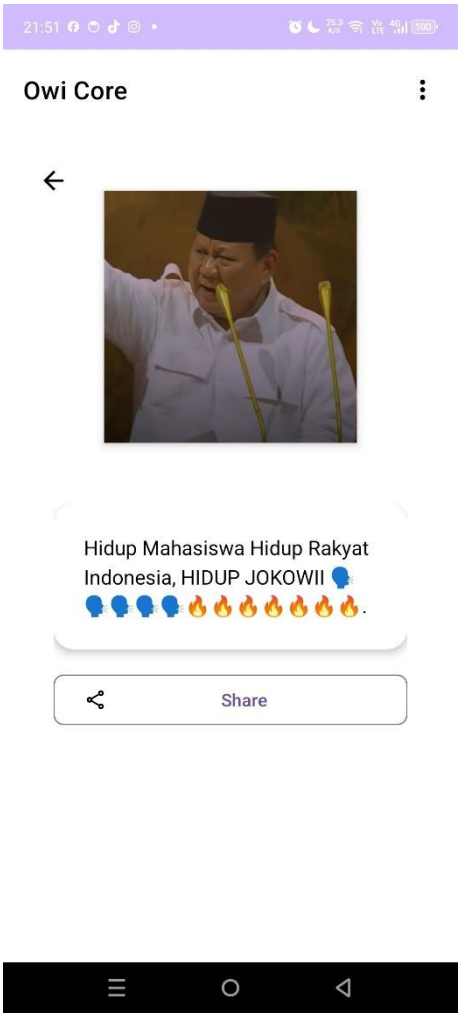
```
35      </fragment>
36
37  </navigation>
```

Tabel 3. 10. Source Code nav_graph.xml

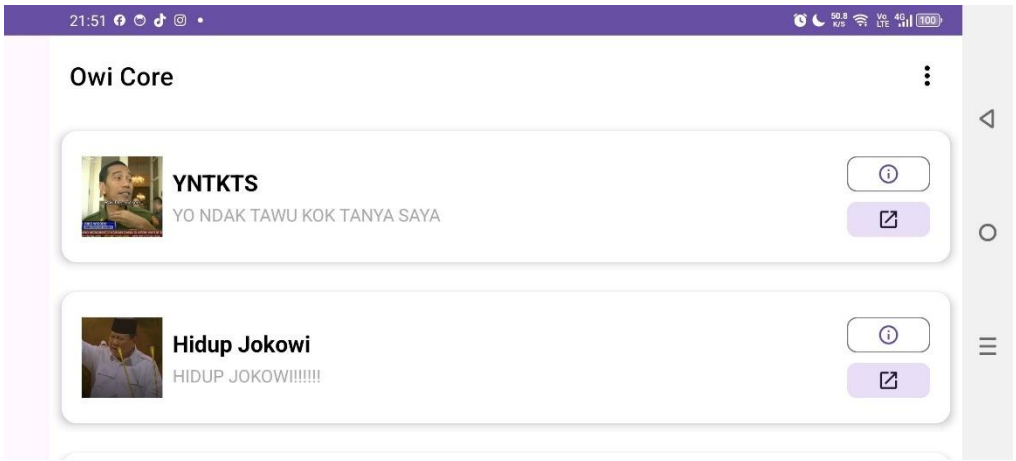
B. Output Program



Gambar 3. 3. Output Home Fragment



Gambar 3. 4. Output Detail Fragment



Gambar 1 Output jika horizontal

C. Pembahasan

1. MainActivity.kt

Pada modul kali ini MainActivity hanya digunakan sebagai wadah untuk fragment-fragment yang ada, sehingga kode pada MainActivity berfokus hanya pada instalasi splash screen, dan juga karena start destination sudah di set pada MyFragment, jadi secara kesuluran MainActivtiy ada hanya sebagai pelengkap dari android manifest itu sendiri.

2. DetailFragment.kt

Overall kegunaan fragment ini hanya sebagai penunjuk detail saja, pada filenya logic yang digunakan hanya menerima data yang di passing oleh adapter seperti pada line [28]-[31] , dan melakukan pergantian image atau ui dari fragment_detail seperti pada line [33]-[38] saja, dan juga beberapa button untuk share, dan kembali pada line [40]-[50].

3. MyAdapter.kt

Oke, Logic dari RecyclerView dimulai pada file ini, dari baris [9] hingga [12], di mana terdapat inisialisasi beberapa parameter, yaitu listCharacter, onYTClick, dan onDetailClick. Parameter listCharacter merupakan data utama yang akan ditampilkan dalam RecyclerView. Parameter onYTClick dan onDetailClick adalah fungsi lambda yang akan dieksekusi ketika tombol tertentu ditekan.

Selanjutnya, pada baris [14] hingga [22], terdapat struktur utama dari RecyclerView Adapter. Pada bagian ini, kelas ListViewHolder digunakan untuk mendefinisikan bahwa setiap elemen yang ditampilkan merupakan bagian dari RecyclerView. Kelas ini menerima parameter berupa ItemLayoutBinding, yang menghubungkan layout XML dengan kode Kotlin.

Fungsi onCreateViewHolder digunakan untuk membuat tampilan item menggunakan LayoutInflater, dengan menghubungkannya ke

ItemLayoutBinding. Ini berarti adapter akan membuat tampilan sesuai template yang sudah didefinisikan di file XML.

Fungsi getItemCount mengembalikan jumlah item yang ada dalam listCharacter, yang berarti menyesuaikan jumlah data yang akan ditampilkan dalam daftar.

Pada bagian akhir, yaitu baris [23] sampai selesai, terdapat fungsi onBindViewHolder. Fungsi ini bertanggung jawab untuk binding data ke dalam tampilan. Artinya, setiap elemen seperti textTitle, textDescription, dan itemImage akan diisi dengan data dari objek yang sesuai di dalam listCharacter. Selain itu, fungsi ini juga menetapkan aksi ketika tombol btnWebsite dan btnDetails ditekan, dengan menjalankan fungsi onYTClick dan onDetailClick yang sudah didefinisikan sebelumnya.

4. MyFragment.kt

Oke, pada file ini, MyFragment merupakan fragment pertama yang akan dijumpai saat membuka aplikasi, karena sudah diatur di nav_graph. Pada baris 17 hingga 21 merupakan template (sering disebut boilerplate) untuk inisialisasi binding dan adapter.

Selanjutnya, pada baris 23 sampai 32 terdapat fungsi onCreateView. Pertama, fungsi ini akan melakukan clear terhadap list yang sudah ada. Setelah itu, data baru akan dimasukkan ke dalam listCharacter, dan kemudian akan memanggil fungsi setupRecyclerView.

Pada baris 34 sampai 52 terdapat logika dari tombol-tombol yang ada. Pada onYTClick, jika tombol ditekan, maka akan dilakukan navigasi ke link yang telah disediakan menggunakan Intent. Sementara itu, pada onDetailClick, data akan dipassing ke DetailFragment menggunakan MyFragmentDirections.

Kemudian, terdapat binding.rvCharacter yang digunakan untuk menerapkan RecyclerView, dengan menggunakan LinearLayoutManager dan menetapkan adapter.

Pada baris 55 hingga 74, terdapat fungsi getListCharacter yang digunakan untuk menjalankan RecyclerView itu sendiri. Fungsi ini mengambil data dari resources, seperti array nama, link, subtext, photo, dan detail, lalu dimasukkan ke dalam ArrayList bertipe MyData. Setelah selesai, TypedArray akan di-recycle.

Terakhir, pada fungsi onDestroyView, dilakukan pembersihan terhadap objek binding agar tidak terjadi memory leak.

5. MyData.kt

Wadah untuk melakukan pendefinisian tipe variable yang digunakan, serta penyimpanan data itu sendiri

6. activity_main.xml

File ini merupakan layout utama yang digunakan oleh MainActivity. Layout ini menggunakan ConstraintLayout sebagai elemen root, yang memungkinkan setiap elemen diatur secara fleksibel berdasarkan hubungan antar elemen atau terhadap parent-nya.

Pertama, terdapat MaterialToolbar dari com.google.android.material.appbar.MaterialToolbar, yang berfungsi sebagai app bar atau toolbar utama. Toolbar ini memiliki lebar match_parent dan tinggi berdasarkan atribut ?attr/actionBarSize. Latar belakangnya berwarna putih dan diberi elevation sebesar 4dp agar terlihat bayangan (shadow) pada perangkat dengan material design. Judul dari toolbar ini adalah "Owi Core", dengan warna teks hitam dan gaya teks sesuai dengan TextAppearance.MaterialComponents.Headline6.

Di dalam MaterialToolbar, terdapat satu buah ImageView dengan id btn_menu. Elemen ini memiliki ukuran lebar dan tinggi 24dp, dengan posisi gravitasi ke end (kanan), serta diberi margin end sebesar 16dp. Ikon yang digunakan diambil dari drawable/ic_more_vert, dan diberi warna menggunakan atribut tint hitam.

Setelah itu, terdapat FragmentContainerView dari androidx.fragment.app.FragmentContainerView, yang digunakan sebagai wadah untuk menampilkan fragment yang aktif. Elemen ini memiliki id nav_host_fragment dan menggunakan NavHostFragment sebagai kelas utama yang menjalankan sistem navigasi. Ukuran lebar dan tingginya diatur menggunakan 0dp karena akan disesuaikan oleh constraint. Komponen ini berada di bawah toolbar, dan dikaitkan ke semua sisi parent untuk memenuhi seluruh layar yang tersisa. Properti app:navGraph menunjukkan nav_graph yang digunakan untuk mengatur alur navigasi antar fragment, dan app:defaultNavHost disetel ke true agar fragment ini bertindak sebagai host utama navigasi.

7. fragment_detail.xml

Layout ini menggunakan NestedScrollView dari androidx.core.widget.NestedScrollView sebagai elemen utama. Elemen ini memungkinkan konten yang lebih panjang dari layar untuk digulir secara vertikal. Properti fillViewport disetel ke true agar isi konten mengisi seluruh area tampilan saat belum bisa digulir.

Di dalamnya terdapat ConstraintLayout yang berfungsi untuk mengatur posisi setiap elemen UI secara fleksibel. Layout ini memiliki padding sebesar 16dp dan tinggi wrap_content, sehingga akan menyesuaikan dengan isi yang dimuat.

Elemen pertama adalah ImageButton dengan id btn_back. Tombol ini berfungsi sebagai tombol kembali. Lebar dan tingginya 48dp, tidak memiliki latar belakang tetap (borderless) karena menggunakan ?selectableItemBackgroundBorderless, dan menggunakan ikon dari drawable/ic_arrow_back. Tombol ini diletakkan di kiri atas dengan menggunakan layout_constraintStart_toStartOf dan layout_constraintTop_toTopOf parent. Warna ikonnya diatur menggunakan app:tint hitam.

Berikutnya adalah ShapeableImageView dari com.google.android.material.imageview.ShapeableImageView yang berfungsi untuk menampilkan gambar utama atau gambar detail. Elemen ini memiliki ukuran 200dp x 200dp, dengan scaleType centerCrop agar gambar terpotong sesuai proporsi tampilan. Gambar ini ditempatkan di tengah horizontal dan di bagian atas layout, dengan margin atas sebesar 32dp dan bayangan sebesar 4dp melalui elevation. Selain itu, elemen ini diberi properti transitionName yaitu shared_image_container untuk mendukung animasi transisi antar fragment.

Setelah itu terdapat LinearLayout yang orientasinya vertikal dan posisinya berada di bawah detail_image. Elemen ini memiliki padding sebesar 24dp dan margin atas 24dp. Di dalamnya terdapat dua elemen utama: CardView dan LinearLayout horizontal.

Pertama, CardView dari androidx.cardview.widget.CardView digunakan untuk menampilkan deskripsi dalam bentuk TextView. Kartu ini memiliki sudut membulat sebesar 16dp, bayangan sebesar 4dp, dan latar belakang putih. Di dalamnya, TextView dengan id detail_description menampilkan teks deskripsi dengan ukuran huruf 16sp, jarak antar baris sebesar 1.2, warna teks hitam, dan padding 24dp di semua sisi.

Kedua, terdapat LinearLayout horizontal yang digunakan untuk menempatkan tombol di bagian bawah konten. Elemen ini memiliki orientasi horizontal dan properti gravity disetel ke center_horizontal untuk memusatkan isinya. Properti spacing diatur sebesar 16dp, walaupun atribut ini tidak berlaku langsung pada LinearLayout standar dan biasanya perlu ditangani melalui margin antar elemen secara manual atau menggunakan spacing dari library tambahan.

Di dalamnya terdapat MaterialButton dari com.google.android.material.button.MaterialButton dengan id btn_share. Tombol ini menggunakan gaya Widget.Material3.Button.OutlinedButton, memiliki lebar match_parent, tinggi 48dp, dan menampilkan teks "Share" serta ikon dari drawable/ic_share. Ikon tidak diberi warna tambahan karena iconTint disetel ke null. Sudut tombol dibulatkan dengan cornerRadius sebesar 8dp.

Secara keseluruhan, layout ini dirancang untuk menampilkan halaman detail dengan gambar utama di atas, deskripsi di tengah, dan tombol aksi di bagian bawah, semuanya dalam struktur yang dapat digulir secara vertikal.

8. fragment_my.xml

File layout ini menggunakan ConstraintLayout dari androidx.constraintlayout.widget.ConstraintLayout sebagai elemen utama. Layout ini memiliki lebar dan tinggi match_parent, artinya akan mengisi seluruh ruang dari layar. Properti tools:context diatur ke .MyFragment, yang memberi petunjuk kepada Android Studio bahwa layout ini digunakan oleh kelas MyFragment. Atribut ini hanya berfungsi saat preview, tidak memengaruhi saat runtime.

Di dalamnya hanya terdapat satu komponen utama yaitu RecyclerView dari androidx.recyclerview.widget.RecyclerView dengan id rv_character. Komponen ini bertugas untuk menampilkan daftar data dalam bentuk list yang bisa digulir. Ukuran lebar dan tingginya diatur ke 0dp karena akan diatur menggunakan constraint. Properti layout_constraintTop_toTopOf, layout_constraintBottom_toBottomOf, layout_constraintStart_toStartOf, dan layout_constraintEnd_toEndOf digunakan untuk mengaitkan semua sisi RecyclerView ke parent-nya, yaitu ConstraintLayout, sehingga elemen ini akan memenuhi seluruh ruang yang tersedia.

Layout ini bersifat minimal dan hanya bertugas menyediakan wadah untuk menampilkan daftar item dalam fragment. Semua logika pemrosesan data dan penanganan klik dilakukan di kelas MyFragment.

9. Item_layout.xml

Layout ini menggunakan CardView dari androidx.cardview.widget.CardView sebagai elemen utama, yang berfungsi untuk membungkus setiap item dalam daftar agar memiliki tampilan seperti kartu. Atribut layout_width diatur ke match_parent dan layout_height ke wrap_content, artinya kartu akan mengisi lebar penuh tapi tingginya akan menyesuaikan kontennya. Properti cardCornerRadius diatur ke 12dp agar memiliki sudut melengkung, cardElevation sebesar 4dp untuk memberi efek bayangan, dan cardUseCompatPadding digunakan agar padding kompatibel dengan perangkat lama.

Di dalam CardView terdapat ConstraintLayout, yang bertugas untuk mengatur posisi elemen-elemen di dalam kartu secara fleksibel. Layout ini memiliki padding 16dp agar konten tidak menempel langsung ke tepi kartu.

Elemen pertama adalah ImageView dengan id item_image. Gambar ini memiliki ukuran tetap 64dp x 64dp, dengan scaleType diatur ke centerCrop agar gambar memenuhi area tanpa merusak rasio. Gambar ini diposisikan secara vertikal di tengah dan di sisi kiri kartu menggunakan constraint ke parent.

Berikutnya adalah LinearLayout yang menampung dua TextView, yaitu text_title dan text_description. Layout ini diletakkan di sebelah kanan gambar menggunakan constraint ke id item_image, dan di sebelah kiri tombol-tombol menggunakan constraint ke button_group. Layout ini disusun secara vertikal.

text_title berfungsi menampilkan judul item, memiliki ukuran teks 18sp, maksimal 1 baris dan akan terpotong dengan ellipsize jika terlalu panjang. text_description menampilkan deskripsi tambahan dengan ukuran teks 14sp, maksimal 2 baris.

Terakhir, terdapat LinearLayout lain dengan id button_group yang menampung dua tombol dari MaterialButton. Layout ini disusun secara vertikal, dan diletakkan di paling kanan kartu. Tombol pertama adalah btn_details, menggunakan style OutlinedButton dan menampilkan ikon ic_info_outline. Tombol kedua adalah btn_website, menggunakan style TonalButton dengan ikon ic_open_in_browser. Kedua tombol ini memiliki tinggi tetap 36dp dan lebar minimum 64dp. Ikon berada di sebelah kiri teks tombol sesuai dengan iconGravity yang diatur ke textStart.

Secara keseluruhan, layout ini dirancang untuk menampilkan item dalam bentuk kartu yang berisi gambar, judul, deskripsi, dan dua tombol aksi.

10. nav_graph.xml

file navigation graph yang digunakan untuk mengatur alur navigasi antar fragment dalam aplikasi. Root tag-nya adalah navigation dari androidx.navigation, dan berfungsi sebagai container dari seluruh rute navigasi. Atribut app:startDestination diatur ke @id/myFragment, artinya aplikasi akan membuka myFragment sebagai tampilan awal.

Di dalamnya terdapat dua elemen fragment.

Fragment pertama memiliki id @+id/myFragment, dengan atribut android:name menunjuk ke com.example.myrecylerview.MyFragment, yaitu kelas yang akan digunakan saat rute ini dipanggil. Label-nya diatur menjadi MyFragment, dan di dalamnya terdapat elemen action.

Elemen action berfungsi untuk mendeklarasikan arah navigasi dari satu fragment ke fragment lain. Dalam hal ini, action_myFragment_to_detailFragment menunjuk ke @id/detailFragment,

yang berarti ketika aksi ini dipanggil, aplikasi akan berpindah dari MyFragment ke DetailFragment.

Fragment kedua adalah detailFragment, dengan atribut android:name menunjuk ke com.example.myrecylerview.DetailFragment. Label-nya diatur menjadi Detail.

Di dalam detailFragment, terdapat tiga elemen argument. Elemen ini digunakan untuk mendeklarasikan argumen yang perlu dikirimkan saat navigasi ke detailFragment dilakukan:

- extraPhoto, dengan tipe integer
- extraLink, dengan tipe string
- extraDetail, dengan tipe string

Argumen-argumen ini digunakan agar data dapat dipassing dari MyFragment ke DetailFragment, seperti gambar, deskripsi detail, dan link eksternal.

SOAL 2

Menurut saya kenapa masih banyak yang memakai RecyclerView meski kode-nya lebih panjang karena alasan praktis dan teknis. Pertama, backward compatibility dan banyak aplikasi besar masih mengandalkan RecyclerView (XML) untuk daftar data besar karena migrasi penuh ke Compose sangat kompleks. Selain itu, RecyclerView menyediakan kontrol tingkat rendah yang lebih besar (misal LayoutManager kustom, animasi item kompleks, dan tipe tampilan item beragam) yang belum sepenuhnya didukung oleh LazyColumn. developer juga sering melakukan adopsi bertahap atau hybrid: Compose dirancang interoperabel dengan UI lama, sehingga bagian aplikasi yang kompleks tetap menggunakan RecyclerView sambil menulis fitur baru dengan LazyColumn. Terakhir, RecyclerView sudah sangat matang (banyak library pihak ketiga dan komunitas luas), sehingga komponen ini tetap menjadi pilihan andal di aplikasi mapan

MODUL 4 : VIEWMODEL AND DEBUGGING

SOAL 1

Soal Praktikum:

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

1. MainActivity.kt

```
1  package com.example.myrecylerview
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.appcompat.app.AppCompatActivity
6  import androidx.appcompat.app.AppCompatDelegate
```

```
7  import
   androidx.core.splashscreen.SplashScreen.Companion.installSpla
   shScreen
8  import
   com.example.myrecylerview.databinding.ActivityMainBinding
9
10 class MainActivity : AppCompatActivity() {
11
12     private lateinit var binding: ActivityMainBinding
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15
16 AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_
   NIGHT_NO)
17         installSplashScreen()
18
19         super.onCreate(savedInstanceState)
20         binding = ActivityMainBinding.inflate(layoutInflater)
           setContentView(binding.root)
21
22         binding.btnMenu.setOnClickListener {
23             Toast.makeText(this, "This Button is used for
24 Decoration", Toast.LENGTH_SHORT).show()
           }
25     }
26 }
```

Tabel 4. 1. Source Code MainActivity.kt

2. DetailFragment.kt

```
1  package com.example.myrecylerview
2
3  import android.content.Intent
4  import android.os.Bundle
5  import android.view.LayoutInflater
```

```
6  import android.view.View
7  import android.view.ViewGroup
8  import androidx.fragment.app.Fragment
9  import androidx.navigation.fragment.findNavController
   import
10 com.example.myrecylerview.databinding.FragmentDetailBinding
11
12 class DetailFragment : Fragment() {
13
14     private var _binding: FragmentDetailBinding? = null
15     private val binding get() = _binding!!
16
17     override fun onCreateView(
18         inflater: LayoutInflater, container: ViewGroup?,
19         savedInstanceState: Bundle?
20     ): View {
           _binding = FragmentDetailBinding.inflate(inflater,
21 container, false)
22         return binding.root
23     }
24
25     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
26         super.onViewCreated(view, savedInstanceState)
27
28         val args =
   DetailFragmentArgs.fromBundle(requireArguments())
29         val photo = args.extraPhoto
30         val link = args.extraLink
31         val detail = args.extraDetail
32
33         binding.detailImage.setImageResource(photo)
34         binding.detailDescription.text = detail
35
36         binding.btnBack.setOnClickListener {
```

```
37              findNavController().navigateUp()
38          }
39
40          binding.btnShare.setOnClickListener {
41              val shareText = buildString {
42                  append("Check this out!\n\n$detail\n\nMore
    info: $link")
43              }
44
45              val shareIntent =
    Intent(Intent.ACTION_SEND).apply {
46                  type = "text/plain"
47                  putExtra(Intent.EXTRA_TEXT, shareText)
48              }
49
50              startActivity(Intent.createChooser(shareIntent,
    "Share via"))
51          }
52      }
53
54      override fun onDestroyView() {
55          super.onDestroyView()
56          _binding = null
57      }
58  }
```

Tabel 4. 2. Source Code DetailFragment.kt

3. MyAdapter.kt

```
1  package com.example.myrecylerview
2
3  import android.view.LayoutInflater
4  import android.view.ViewGroup
5  import androidx.recyclerview.widget.RecyclerView
6  import com.example.myrecylerview.databinding.ItemLayoutBinding
```

```
7
8   class MyAdapter(
9       private val onYTClick: (String) -> Unit,
10      private val onDetailClick: (String, Int, String) -> Unit
11  ) : RecyclerView.Adapter<MyAdapter.ListViewHolder>() {
12
13      private val items = ArrayList<MyData>()
14
15      fun submitList(newList: List<MyData>) {
16          items.clear()
17          items.addAll(newList)
18          notifyDataSetChanged()
19      }
20
21      class ListViewHolder(val binding: ItemLayoutBinding) :
        RecyclerView.ViewHolder(binding.root)
22
23      override fun onCreateViewHolder(parent: ViewGroup,
        viewType: Int): ListViewHolder {
24          val binding =
        ItemLayoutBinding.inflate(LayoutInflater.from(parent.context),
        parent, false)
25          return ListViewHolder(binding)
26      }
27
28      override fun getItemCount(): Int = items.size
29
30      override fun onBindViewHolder(holder: ListViewHolder,
        position: Int) {
31          val (name, link, photo, detail, subtext) =
        items[position]
32          with(holder.binding) {
33              textTitle.text = name
34              textDescription.text = subtext
35              itemImage.setImageResource(photo)
```

```
36              btnWebsite.setOnClickListener { onYTClick(link) }
37              btnDetails.setOnClickListener
    { onDetailClick(detail, photo, link) }
38          }
39      }
40  }
```

Tabel 4. 3. Source Code MyAdapter.kt

4. MyFragment.kt

```
1   package com.example.myrecylerview
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.fragment.app.Fragment
10  import androidx.lifecycle.ViewModelProvider
11  import androidx.lifecycle.lifecycleScope
12  import androidx.navigation.fragment.findNavController
13  import androidx.recyclerview.widget.LinearLayoutManager
14  import com.example.myrecylerview.databinding.FragmentMyBinding


15
16  class MyFragment : Fragment() {
17
18      private var _binding: FragmentMyBinding? = null
19      private val binding get() = _binding!!
20
21      private lateinit var viewModel: MyViewModel
22      private lateinit var characterAdapter: MyAdapter
23
```

```
24      override fun onCreateView(
25          inflater: LayoutInflater, container: ViewGroup?,
26          savedInstanceState: Bundle?
27      ): View {
28          _binding = FragmentMyBinding.inflate(inflater, container, fal
29          return binding.root
        }
30
31      override fun onViewCreated(view: View, savedInstanceState: Bundle
32          super.onViewCreated(view, savedInstanceState)
33
            viewModel = ViewModelProvider(
34              this,
35              MyViewModelFactory(requireActivity().application)
36          )[MyViewModel::class.java]
37
38          characterAdapter = MyAdapter(
                onYTClick = { link -> viewModel.onYTClick(link) },
39              onDetailClick = { detail, photo, link ->
40  viewModel.onDetailClick(detail, photo, link) }
41          )
42
43          binding.rvCharacter.apply {
                layoutManager = LinearLayoutManager(requireContext())
44              adapter = characterAdapter
45              setHasFixedSize(true)
46          }
47
            lifecycleScope.launchWhenStarted {
48              viewModel.characterList.collect { list ->
49                  characterAdapter.submitList(list)
50              }
51          }
52
53          lifecycleScope.launchWhenStarted {
```

```
54              viewModel.uiEvent.collect { event ->
55                  when (event) {
56                      is MyViewModel.UiEvent.OpenYouTube -> {
57                          val intent = Intent(Intent.ACTION_VIEW,
58  Uri.parse(event.link))
59                          startActivity(intent)
60                      }
61                      is MyViewModel.UiEvent.NavigateToDetail -> {
62                          val action = MyFragmentDirections
63                              .actionMyFragmentToDetailFragment(event.p
    event.link, event.detail)
64                          findNavController().navigate(action)
65                      }
66                      null -> Unit
                    }
67                  viewModel.clearEvent()
68              }
69          }

70          viewModel.loadCharacterList()
71      }
72
73      override fun onDestroyView() {
74          super.onDestroyView()
75          _binding = null
76      }
77  }
78
79
80
81
82
83
84
```

Tabel 4. 4. Source Code Soal MyFragment.kt

5. MyData.kt

```
1  package com.example.myrecylerview
2
3  import android.os.Parcelable
4  import kotlinx.parcelize.Parcelize
5
6  @Parcelize
7  data class MyData(
8      val name: String,
9      val link: String,
10     val photo: Int,
11     val detail: String,
12     val subtext: String
13 ):Parcelable
```

Tabel 4. 5. Source Code Soal MyData.kt

6. MyViewModel.kt

```
1  package com.example.myrecylerview
2
3  import android.app.Application
4  import android.util.Log
5  import androidx.lifecycle.AndroidViewModel
6  import kotlinx.coroutines.flow.MutableStateFlow
7  import kotlinx.coroutines.flow.StateFlow
8
9  class MyViewModel(application: Application) :
   AndroidViewModel(application) {
10
11     private val _characterList =
12 MutableStateFlow<List<MyData>>(emptyList())
       val characterList: StateFlow<List<MyData>> =
```

```
13  _characterList

14      private val _uiEvent = MutableStateFlow<UiEvent?>(null)
15      val uiEvent: StateFlow<UiEvent?> = _uiEvent
16
17      fun loadCharacterList() {
18          val context = getApplication<Application>()
19          val resources = context.resources
20
21          val dataName =
    resources.getStringArray(R.array.data_name)
22          val dataLink =
    resources.getStringArray(R.array.data_link)
23          val dataSubtext =
    resources.getStringArray(R.array.data_subtext)
24          val dataPhoto =
    resources.obtainTypedArray(R.array.data_photo)
25          val dataDetail =
    resources.getStringArray(R.array.data_detail)
26
27          val listCharacter = List(dataName.size) { i ->
28              MyData(
29                  name = dataName[i],
30                  link = dataLink[i],
31                  subtext = dataSubtext[i],
32                  detail = dataDetail[i],
33                  photo = dataPhoto.getResourceId(i, -1)
34              )
35          }
36
37          dataPhoto.recycle()
38          _characterList.value = listCharacter
39
40          Log.d("MyViewModel", "Loaded ${listCharacter.size}
    characters: $listCharacter")
```

```
41      }
42
43      fun onYTClick(link: String) {
44          Log.d("MyViewModel", "YouTube button clicked with
    link: $link")
45          _uiEvent.value = UiEvent.OpenYouTube(link)
46      }
47
48      fun onDetailClick(detail: String, photo: Int, link:
    String) {
49          Log.d("MyViewModel", "Detail button clicked with data
    -> detail: $detail, photo: $photo, link: $link")
50          _uiEvent.value = UiEvent.NavigateToDetail(photo,
    link, detail)
51      }
52
53      fun clearEvent() {
54          _uiEvent.value = null
55      }
56
57      sealed class UiEvent {
58          data class OpenYouTube(val link: String) : UiEvent()
59          data class NavigateToDetail(val photo: Int, val link:
    String, val detail: String) : UiEvent()
60      }
61  }
```

Tabel 4. 6. Source Code MyViewModel.kt

7. MyViewModelFactory.kt

```
1   package com.example.myrecylerview
2
3   import android.app.Application
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
```

```
6
7   class MyViewModelFactory(private val application:
    Application) : ViewModelProvider.Factory {
8       override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
9           if
    (modelClass.isAssignableFrom(MyViewModel::class.java)) {
10              return MyViewModel(application) as T
11          }
12          throw IllegalArgumentException("Unknown ViewModel
    class")
13      }
14
15  }
```

Tabel 4. 7. Source Code MyViewModelFactory.kt

8. activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@android:color/white"
8       tools:context=".MainActivity">
9
10      <com.google.android.material.appbar.MaterialToolbar
11          android:id="@+id/toolbar"
12          android:layout_width="match_parent"
13          android:layout_height="?attr/actionBarSize"
14          android:background="@android:color/white"
15          android:elevation="4dp"
16          app:layout_constraintEnd_toEndOf="parent"
```

```
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:title="Owi Core"
20          app:titleTextColor="@android:color/black"
21
22  app:titleTextAppearance="@style/TextAppearance.MaterialComponents.Hea
23
24          <ImageView
25              android:id="@+id/btn_menu"
26              android:layout_width="24dp"
27              android:layout_height="24dp"
28              android:layout_gravity="end"
29              android:layout_marginEnd="16dp"
30              android:src="@drawable/ic_more_vert"
31              app:tint="@android:color/black" />
32
33      </com.google.android.material.appbar.MaterialToolbar>
34
35      <androidx.fragment.app.FragmentContainerView
36          android:id="@+id/nav_host_fragment"
37          android:name="androidx.navigation.fragment.NavHostFragment"
38          android:layout_width="0dp"
39          android:layout_height="0dp"
40          app:layout_constraintTop_toBottomOf="@id/toolbar"
41          app:layout_constraintBottom_toBottomOf="parent"
42          app:layout_constraintStart_toStartOf="parent"
43          app:layout_constraintEnd_toEndOf="parent"
44          app:navGraph="@navigation/nav_graph"
45          app:defaultNavHost="true" />
46
47  </androidx.constraintlayout.widget.ConstraintLayout>
48
49
```

Tabel 4. 8. Source Code activity_main.xml

9. fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.core.widget.NestedScrollView

3   xmlns:android="http://schemas.android.com/apk/res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:fillViewport="true">
9
10      <androidx.constraintlayout.widget.ConstraintLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:padding="16dp">
14
15          <ImageButton
16              android:id="@+id/btn_back"
17              android:layout_width="48dp"
18              android:layout_height="48dp"
19
20  android:background="?selectableItemBackgroundBorderless"
21              android:src="@drawable/ic_arrow_back"
22              app:layout_constraintStart_toStartOf="parent"
23              app:layout_constraintTop_toTopOf="parent"
24              app:tint="@android:color/black" />
25
26
27  <com.google.android.material.imageview.ShapeableImageView
28              android:id="@+id/detail_image"
29              android:layout_width="200dp"
30              android:layout_height="200dp"
31              android:scaleType="centerCrop"
32              app:layout_constraintEnd_toEndOf="parent"
33              app:layout_constraintStart_toStartOf="parent"
```

```
34              app:layout_constraintTop_toTopOf="parent"
35              app:layout_constraintVertical_bias="0"
36              android:layout_marginTop="32dp"
37              android:elevation="4dp"
38
39  android:transitionName="shared_image_container"/>
40
41          <LinearLayout
42              android:layout_width="match_parent"
43              android:layout_height="wrap_content"
44              android:orientation="vertical"
45              android:padding="24dp"
46              app:layout_constraintEnd_toEndOf="parent"
47              app:layout_constraintStart_toStartOf="parent"
48
49  app:layout_constraintTop_toBottomOf="@id/detail_image"
50              android:layout_marginTop="24dp">
51
52              <androidx.cardview.widget.CardView
53                  android:layout_width="match_parent"
54                  android:layout_height="wrap_content"
55                  app:cardCornerRadius="16dp"
56                  app:cardElevation="4dp"
57
58  app:cardBackgroundColor="@android:color/white"
59                  android:layout_marginBottom="16dp">
60
61                  <TextView
62                      android:id="@+id/detail_description"
63                      android:layout_width="match_parent"
64                      android:layout_height="wrap_content"
65                      android:textSize="16sp"
66                      android:lineSpacingMultiplier="1.2"
67                      android:textColor="@android:color/black"
68                      android:padding="24dp"
```

```
69                      tools:text="Lorem ipsum dolor sit amet,
70  consectetur adipiscing elit. Sed do eiusmod tempor
71  incididunt ut labore et dolore magna aliqua. Ut enim ad
72  minim veniam, quis nostrud exercitation ullamco laboris nisi
73  ut aliquip ex ea commodo consequat."/>
74
75              </androidx.cardview.widget.CardView>
76
77              <LinearLayout
78                  android:layout_width="match_parent"
79                  android:layout_height="wrap_content"
80                  android:orientation="horizontal"
81                  android:gravity="center_horizontal"
82                  android:spacing="16dp">
83
84
85  <com.google.android.material.button.MaterialButton
86                      android:id="@+id/btn_share"
87
88  style="@style/Widget.Material3.Button.OutlinedButton"
89                      android:layout_width="match_parent"
90                      android:layout_height="48dp"
91                      android:text="Share"
92                      app:icon="@drawable/ic_share"
93                      app:iconTint="@null"
94                      app:cornerRadius="8dp"/>
95
96              </LinearLayout>
97
98          </LinearLayout>
99
100     </androidx.constraintlayout.widget.ConstraintLayout>
101
102 </androidx.core.widget.NestedScrollView>
```

Tabel 4. 9. Source Code fragment_detail.xml

10. fragment_my.xml

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MyFragment">


    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_character"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 4. 10. Source Code fragment_my.xml

11. item_layout.xml

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


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:layout_width="match_parent"
    android:layout_height="wrap_content"
```

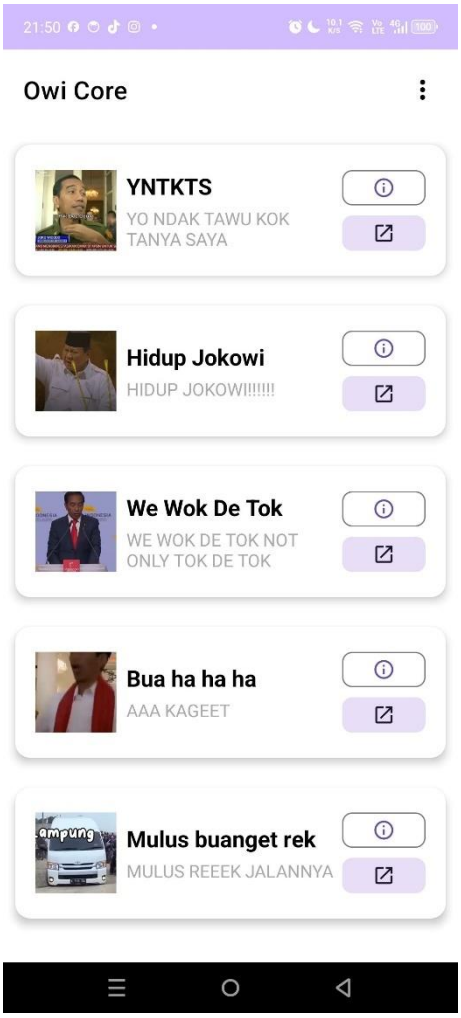
```
8       android:layout_margin="2dp"
9       app:cardCornerRadius="12dp"
10      app:cardElevation="4dp"
11      app:cardUseCompatPadding="true">
12
13      <androidx.constraintlayout.widget.ConstraintLayout
14          android:layout_width="match_parent"
15          android:layout_height="wrap_content"
16          android:padding="16dp">
17
18          <ImageView
19              android:src="@drawable/pakdhe_1"
20              android:id="@+id/item_image"
21              android:layout_width="64dp"
22              android:layout_height="64dp"
23              android:scaleType="centerCrop"
24              app:layout_constraintBottom_toBottomOf="parent"
25              app:layout_constraintStart_toStartOf="parent"
26              app:layout_constraintTop_toTopOf="parent"
27              android:layout_marginEnd="16dp"/>
28
29          <LinearLayout
30              android:layout_width="0dp"
31              android:layout_height="wrap_content"
32              android:orientation="vertical"
33
34  app:layout_constraintBottom_toBottomOf="@id/item_image"
35
36  app:layout_constraintEnd_toStartOf="@+id/button_group"
37
38  app:layout_constraintStart_toEndOf="@id/item_image"
39
40  app:layout_constraintTop_toTopOf="@id/item_image">
41
42              <TextView
```

```
                android:id="@+id/text_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:ellipsize="end"
                android:maxLines="1"
                android:textColor="@android:color/black"
                android:textSize="18sp"
                android:textStyle="bold"
                tools:text="Main Title" />

            <TextView
                android:id="@+id/text_description"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginStart="8dp"
                android:layout_marginTop="4dp"
                android:ellipsize="end"
                android:maxLines="2"

android:textColor="@android:color/darker_gray"
                android:textSize="14sp"
                tools:text="Secondary description text that
can span multiple lines" />
        </LinearLayout>

        <LinearLayout
            android:id="@+id/button_group"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:gravity="end"
            android:spacing="8dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
```

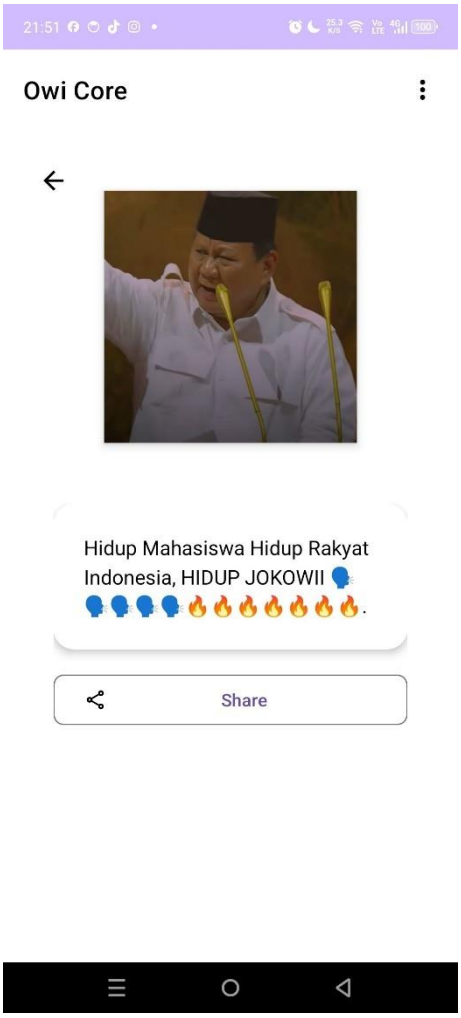
```
78              app:layout_constraintTop_toTopOf="parent">
79
80
81  <com.google.android.material.button.MaterialButton
82                  android:id="@+id/btn_details"
83
84  style="@style/Widget.Material3.Button.OutlinedButton"
85                  android:layout_width="wrap_content"
86                  android:layout_height="36dp"
87                  android:minWidth="64dp"
88                  app:icon="@drawable/ic_info_outline"
89                  app:iconPadding="0dp"
90                  app:iconGravity="textStart"
91                  app:cornerRadius="8dp"/>
92
93
94  <com.google.android.material.button.MaterialButton
95                  android:id="@+id/btn_website"
96
97  style="@style/Widget.Material3.Button.TonalButton"
98                  android:layout_width="wrap_content"
99                  android:layout_height="36dp"
100                 android:minWidth="64dp"
101                 app:icon="@drawable/ic_open_in_browser"
102                 app:iconPadding="0dp"
103                 app:iconGravity="textStart"
104                 app:cornerRadius="8dp"/>
105         </LinearLayout>
106
107     </androidx.constraintlayout.widget.ConstraintLayout>
108 </androidx.cardview.widget.CardView>
```

Tabel 4. 11. Source Code item_layout.xml

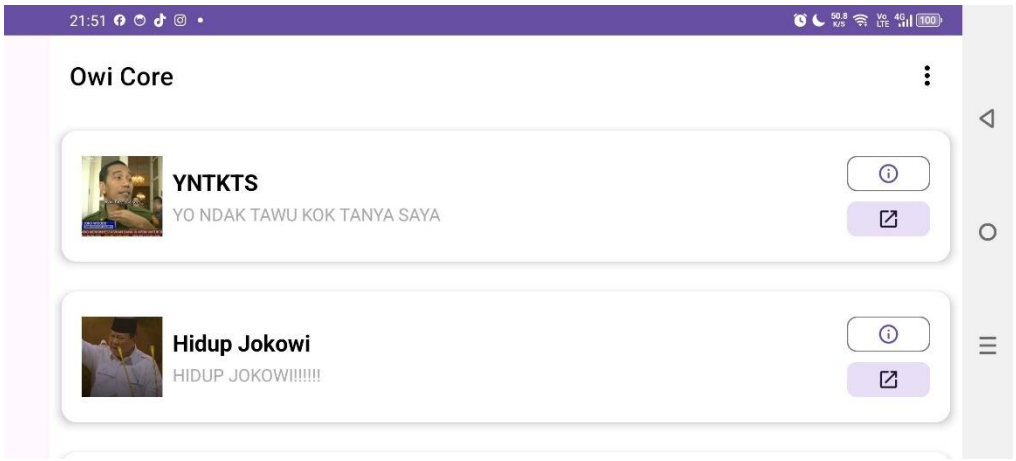
B. Output Program



Gambar 4. 1. Output Home Fragment



Gambar 4. 2. Output Detail Fragment



Gambar 4. 3. Screenshot Output Jika Horizontal

C. Pembahasan

1. MainActivity.kt

Pada modul kali ini MainActivity hanya digunakan sebagai wadah untuk fragment-fragment yang ada, sehingga kode pada MainActivity berfokus hanya pada instalasi splash screen, dan juga karena start destination sudah di set pada MyFragment, jadi secara kesuluran MainActivtiy ada hanya sebagai pelengkap dari android manifest itu sendiri.

2. DetailFragment.kt

Overall kegunaan fragment ini hanya sebagai penunjuk detail saja, pada filenya logic yang digunakan hanya menerima data yang di passing oleh adapter seperti pada line [28]-[31] , dan melakukan pergantian image atau ui dari fragment_detail seperti pada line [33]-[38] saja, dan juga beberapa button untuk share, dan kembali pada line [40]-[50].

3. MyAdapter.kt

Oke, Logic dari RecyclerView dimulai pada file ini, dari baris [9] hingga [12], di mana terdapat inisialisasi beberapa parameter, yaitu listCharacter, onYTClick, dan onDetailClick. Parameter listCharacter merupakan data utama yang akan ditampilkan dalam RecyclerView. Parameter onYTClick dan onDetailClick adalah fungsi lambda yang akan dieksekusi ketika tombol tertentu ditekan.

Selanjutnya, pada baris [14] hingga [22], terdapat struktur utama dari RecyclerView Adapter. Pada bagian ini, kelas ListViewHolder digunakan untuk mendefinisikan bahwa setiap elemen yang ditampilkan merupakan bagian dari RecyclerView. Kelas ini menerima parameter berupa ItemLayoutBinding, yang menghubungkan layout XML dengan kode Kotlin.

Fungsi onCreateViewHolder digunakan untuk membuat tampilan item menggunakan LayoutInflater, dengan menghubungkannya ke

ItemLayoutBinding. Ini berarti adapter akan membuat tampilan sesuai template yang sudah didefinisikan di file XML.

Fungsi getItemCount mengembalikan jumlah item yang ada dalam listCharacter, yang berarti menyesuaikan jumlah data yang akan ditampilkan dalam daftar.

Pada bagian akhir, yaitu baris [23] sampai selesai, terdapat fungsi onBindViewHolder. Fungsi ini bertanggung jawab untuk binding data ke dalam tampilan. Artinya, setiap elemen seperti textTitle, textDescription, dan itemImage akan diisi dengan data dari objek yang sesuai di dalam listCharacter. Selain itu, fungsi ini juga menetapkan aksi ketika tombol btnWebsite dan btnDetails ditekan, dengan menjalankan fungsi onYTClick dan onDetailClick yang sudah didefinisikan sebelumnya.

4. MyFragment.kt

Oke, pada file ini, MyFragment merupakan fragment pertama yang akan dijumpai saat membuka aplikasi, karena sudah diatur di nav_graph. Pada baris 17 hingga 21 merupakan template (sering disebut boilerplate) untuk inisialisasi binding dan adapter.

Selanjutnya, pada baris 23 sampai 32 terdapat fungsi onCreateView. Pertama, fungsi ini akan melakukan clear terhadap list yang sudah ada. Setelah itu, data baru akan dimasukkan ke dalam listCharacter, dan kemudian akan memanggil fungsi setupRecyclerView.

Pada baris 34 sampai 52 terdapat logika dari tombol-tombol yang ada. Pada onYTClick, jika tombol ditekan, maka akan dilakukan navigasi ke link yang telah disediakan menggunakan Intent. Sementara itu, pada onDetailClick, data akan dipassing ke DetailFragment menggunakan MyFragmentDirections.

Kemudian, terdapat binding.rvCharacter yang digunakan untuk menerapkan RecyclerView, dengan menggunakan LinearLayoutManager dan menetapkan adapter.

Pada baris 55 hingga 74, terdapat fungsi getListCharacter yang digunakan untuk menjalankan RecyclerView itu sendiri. Fungsi ini mengambil data dari resources, seperti array nama, link, subtext, photo, dan detail, lalu dimasukkan ke dalam ArrayList bertipe MyData. Setelah selesai, TypedArray akan di-recycle.

Terakhir, pada fungsi onDestroyView, dilakukan pembersihan terhadap objek binding agar tidak terjadi memory leak.

5. MyData.kt

Wadah untuk melakukan pendefinisian tipe variable yang digunakan, serta penyimpanan data itu sendiri

6. MyViewModel.kt

MyViewModel adalah turunan dari AndroidViewModel yang mengelola data dan event UI secara terpisah dari fragment. Di dalamnya, terdapat dua MutableStateFlow: satu untuk menyimpan daftar karakter (_characterList) dan satu lagi untuk event UI tunggal (_uiEvent), masing-masing diekspos sebagai StateFlow hanya-baca agar fragment dapat mengamati perubahan tanpa memodifikasi langsung. Fungsi loadCharacterList() mengambil konteks aplikasi untuk membaca resource array—nama, link YouTube, subteks, detail, dan foto—kemudian membangun List<MyData> dengan memetakan tiap indeks ke objek MyData yang sesuai, setelah itu me-recycle() TypedArray foto untuk mencegah memory leak dan mengubah nilai _characterList, sehingga UI ter-update secara reaktif.

Ketika tombol YouTube ditekan, onYTClick(link) mencatat log dan memancarkan UiEvent.OpenYouTube(link), sementara onDetailClick(detail, photo, link) mencatat log serupa lalu memancarkan UiEvent.NavigateToDetail(photo, link, detail) untuk menginstruksi fragment agar menavigasi ke detail screen melalui Safe Args. Setelah event ditangani, clearEvent() dipanggil untuk mengatur ulang _uiEvent menjadi null, mencegah pengulangan event saat re-collect berlangsung. Dengan pola ini, MyViewModel menjaga UI tetap bersih dari logika pemrosesan data

dan navigasi, memanfaatkan StateFlow yang lifecycle-aware untuk memudahkan pengujian dan menghindari memory leak.

7. MyViewModelFactory.kt

MyViewModelFactory mengimplementasikan ViewModelProvider.Factory untuk menyediakan instans MyViewModel yang memerlukan Application sebagai parameter konstruktor. Ketika create() dipanggil, factory memeriksa apakah modelClass dapat di-assign ke MyViewModel::class.java; jika ya, ia mengembalikan objek baru MyViewModel(application) setelah melakukan cast ke tipe generik T. Jika tidak cocok, factory melempar IllegalArgumentException, sehingga mencegah pembuatan ViewModel yang tak terduga. Kehadiran factory ini memungkinkan fragment atau activity untuk menggunakan ViewModelProvider(this, MyViewModelFactory(requireActivity().application)) sehingga MyViewModel dapat menggunakan context aplikasi dengan benar, sekaligus menjaga integritas tipe ViewModel.

8. activity_main.xml

File ini merupakan layout utama yang digunakan oleh MainActivity. Layout ini menggunakan ConstraintLayout sebagai elemen root, yang memungkinkan setiap elemen diatur secara fleksibel berdasarkan hubungan antar elemen atau terhadap parent-nya.

Pertama, terdapat MaterialToolbar dari com.google.android.material.appbar.MaterialToolbar, yang berfungsi sebagai app bar atau toolbar utama. Toolbar ini memiliki lebar match_parent dan tinggi berdasarkan atribut ?attr/actionBarSize. Latar belakangnya berwarna putih dan diberi elevation sebesar 4dp agar terlihat bayangan (shadow) pada perangkat dengan material design. Judul dari toolbar ini adalah "Owi Core", dengan warna teks hitam dan gaya teks sesuai dengan TextAppearance.MaterialComponents.Headline6.

Di dalam MaterialToolbar, terdapat satu buah ImageView dengan id btn_menu. Elemen ini memiliki ukuran lebar dan tinggi 24dp, dengan posisi gravitasi ke end (kanan), serta diberi margin end sebesar 16dp. Ikon yang digunakan diambil dari drawable/ic_more_vert, dan diberi warna menggunakan atribut tint hitam.

Setelah itu, terdapat FragmentContainerView dari androidx.fragment.app.FragmentContainerView, yang digunakan sebagai wadah untuk menampilkan fragment yang aktif. Elemen ini memiliki id nav_host_fragment dan menggunakan NavHostFragment sebagai kelas utama yang menjalankan sistem navigasi. Ukuran lebar dan tingginya diatur menggunakan 0dp karena akan disesuaikan oleh constraint. Komponen ini berada di bawah toolbar, dan dikaitkan ke semua sisi parent untuk memenuhi seluruh layar yang tersisa. Properti app:navGraph menunjukkan nav_graph yang digunakan untuk mengatur alur navigasi antar fragment, dan app:defaultNavHost disetel ke true agar fragment ini bertindak sebagai host utama navigasi.

9. fragment_detail.xml

Layout ini menggunakan NestedScrollView dari androidx.core.widget.NestedScrollView sebagai elemen utama. Elemen ini memungkinkan konten yang lebih panjang dari layar untuk digulir secara vertikal. Properti fillViewport disetel ke true agar isi konten mengisi seluruh area tampilan saat belum bisa digulir.

Di dalamnya terdapat ConstraintLayout yang berfungsi untuk mengatur posisi setiap elemen UI secara fleksibel. Layout ini memiliki padding sebesar 16dp dan tinggi wrap_content, sehingga akan menyesuaikan dengan isi yang dimuat.

Elemen pertama adalah ImageButton dengan id btn_back. Tombol ini berfungsi sebagai tombol kembali. Lebar dan tingginya 48dp, tidak memiliki latar belakang tetap (borderless) karena menggunakan

?selectableItemBackgroundBorderless, dan menggunakan ikon dari drawable/ic_arrow_back. Tombol ini diletakkan di kiri atas dengan menggunakan layout_constraintStart_toStartOf dan layout_constraintTop_toTopOf parent. Warna ikonnya diatur menggunakan app:tint hitam.

Berikutnya adalah ShapeableImageView dari com.google.android.material.imageview.ShapeableImageView yang berfungsi untuk menampilkan gambar utama atau gambar detail. Elemen ini memiliki ukuran 200dp x 200dp, dengan scaleType centerCrop agar gambar terpotong sesuai proporsi tampilan. Gambar ini ditempatkan di tengah horizontal dan di bagian atas layout, dengan margin atas sebesar 32dp dan bayangan sebesar 4dp melalui elevation. Selain itu, elemen ini diberi properti transitionName yaitu shared_image_container untuk mendukung animasi transisi antar fragment.

Setelah itu terdapat LinearLayout yang orientasinya vertikal dan posisinya berada di bawah detail_image. Elemen ini memiliki padding sebesar 24dp dan margin atas 24dp. Di dalamnya terdapat dua elemen utama: CardView dan LinearLayout horizontal.

Pertama, CardView dari androidx.cardview.widget.CardView digunakan untuk menampilkan deskripsi dalam bentuk TextView. Kartu ini memiliki sudut membulat sebesar 16dp, bayangan sebesar 4dp, dan latar belakang putih. Di dalamnya, TextView dengan id detail_description menampilkan teks deskripsi dengan ukuran huruf 16sp, jarak antar baris sebesar 1.2, warna teks hitam, dan padding 24dp di semua sisi.

Kedua, terdapat LinearLayout horizontal yang digunakan untuk menempatkan tombol di bagian bawah konten. Elemen ini memiliki orientasi horizontal dan properti gravity disetel ke center_horizontal untuk memusatkan isinya. Properti spacing diatur sebesar 16dp, walaupun atribut

ini tidak berlaku langsung pada LinearLayout standar dan biasanya perlu ditangani melalui margin antar elemen secara manual atau menggunakan spacing dari library tambahan.

Di dalamnya terdapat MaterialButton dari com.google.android.material.button.MaterialButton dengan id btn_share. Tombol ini menggunakan gaya Widget.Material3.Button.OutlinedButton, memiliki lebar match_parent, tinggi 48dp, dan menampilkan teks "Share" serta ikon dari drawable/ic_share. Ikon tidak diberi warna tambahan karena iconTint disetel ke null. Sudut tombol dibulatkan dengan cornerRadius sebesar 8dp.

Secara keseluruhan, layout ini dirancang untuk menampilkan halaman detail dengan gambar utama di atas, deskripsi di tengah, dan tombol aksi di bagian bawah, semuanya dalam struktur yang dapat digulir secara vertikal.

10. fragment_my.xml

File layout ini menggunakan ConstraintLayout dari androidx.constraintlayout.widget.ConstraintLayout sebagai elemen utama. Layout ini memiliki lebar dan tinggi match_parent, artinya akan mengisi seluruh ruang dari layar. Properti tools:context diatur ke .MyFragment, yang memberi petunjuk kepada Android Studio bahwa layout ini digunakan oleh kelas MyFragment. Atribut ini hanya berfungsi saat preview, tidak memengaruhi saat runtime.

Di dalamnya hanya terdapat satu komponen utama yaitu RecyclerView dari androidx.recyclerview.widget.RecyclerView dengan id rv_character. Komponen ini bertugas untuk menampilkan daftar data dalam bentuk list yang bisa digulir. Ukuran lebar dan tingginya diatur ke 0dp karena akan diatur menggunakan constraint. Properti layout_constraintTop_toTopOf, layout_constraintBottom_toBottomOf,

layout_constraintStart_toStartOf, dan layout_constraintEnd_toEndOf digunakan untuk mengaitkan semua sisi RecyclerView ke parent-nya, yaitu ConstraintLayout, sehingga elemen ini akan memenuhi seluruh ruang yang tersedia.

Layout ini bersifat minimal dan hanya bertugas menyediakan wadah untuk menampilkan daftar item dalam fragment. Semua logika pemrosesan data dan penanganan klik dilakukan di kelas MyFragment.

11. Item_layout.xml

Layout ini menggunakan CardView dari androidx.cardview.widget.CardView sebagai elemen utama, yang berfungsi untuk membungkus setiap item dalam daftar agar memiliki tampilan seperti kartu. Atribut layout_width diatur ke match_parent dan layout_height ke wrap_content, artinya kartu akan mengisi lebar penuh tapi tingginya akan menyesuaikan kontennya. Properti cardCornerRadius diatur ke 12dp agar memiliki sudut melengkung, cardElevation sebesar 4dp untuk memberi efek bayangan, dan cardUseCompatPadding digunakan agar padding kompatibel dengan perangkat lama.

Di dalam CardView terdapat ConstraintLayout, yang bertugas untuk mengatur posisi elemen-elemen di dalam kartu secara fleksibel. Layout ini memiliki padding 16dp agar konten tidak menempel langsung ke tepi kartu.

Elemen pertama adalah ImageView dengan id item_image. Gambar ini memiliki ukuran tetap 64dp x 64dp, dengan scaleType diatur ke centerCrop agar gambar memenuhi area tanpa merusak rasio. Gambar ini diposisikan secara vertikal di tengah dan di sisi kiri kartu menggunakan constraint ke parent.

Berikutnya adalah LinearLayout yang menampung dua TextView, yaitu text_title dan text_description. Layout ini diletakkan di sebelah kanan

gambar menggunakan constraint ke id item_image, dan di sebelah kiri tombol-tombol menggunakan constraint ke button_group. Layout ini disusun secara vertikal.

text_title berfungsi menampilkan judul item, memiliki ukuran teks 18sp, maksimal 1 baris dan akan terpotong dengan ellipsize jika terlalu panjang. text_description menampilkan deskripsi tambahan dengan ukuran teks 14sp, maksimal 2 baris.

Terakhir, terdapat LinearLayout lain dengan id button_group yang menampung dua tombol dari MaterialButton. Layout ini disusun secara vertikal, dan diletakkan di paling kanan kartu. Tombol pertama adalah btn_details, menggunakan style OutlinedButton dan menampilkan ikon ic_info_outline. Tombol kedua adalah btn_website, menggunakan style TonalButton dengan ikon ic_open_in_browser. Kedua tombol ini memiliki tinggi tetap 36dp dan lebar minimum 64dp. Ikon berada di sebelah kiri teks tombol sesuai dengan iconGravity yang diatur ke textStart.

Secara keseluruhan, layout ini dirancang untuk menampilkan item dalam bentuk kartu yang berisi gambar, judul, deskripsi, dan dua tombol aksi.

12. Penjelasan debugger.

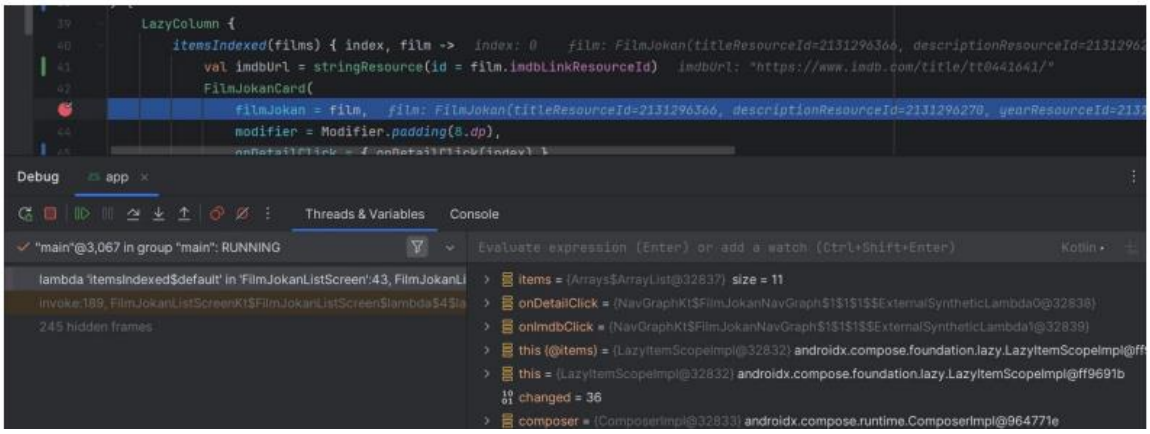
Debugger di Android Studio adalah alat interaktif yang memungkinkan Anda menghentikan eksekusi aplikasi pada titik yang telah ditentukan (breakpoint) untuk memeriksa nilai variabel, tumpukan panggilan (call stack), dan ekspresi yang dipantau (watches) secara real time, sehingga memudahkan proses identifikasi dan perbaikan bug. Untuk menggunakannya, pertama-tama Anda menempatkan breakpoint dengan mengklik margin kiri di baris kode yang ingin dipantau, lalu menjalankan aplikasi dalam mode Debug; eksekusi akan berhenti tepat di breakpoint tersebut. Setelah pause, Anda dapat memeriksa panel Variables untuk

melihat nilai-nilai saat ini, memodifikasi watches, dan mengamati alur program. Tombol “Step Into” (F7) akan memasuki implementasi fungsi atau metode yang dipanggil pada baris saat ini, memungkinkan Anda menelusuri internal fungsi tersebut, sedangkan “Step Over” (F8) akan menjalankan seluruh baris termasuk pemanggilan fungsi tanpa masuk ke dalamnya, kemudian berhenti di baris berikutnya, cocok digunakan ketika Anda percaya fungsi tersebut sudah benar. Jika Anda sudah berada di dalam sebuah fungsi dan ingin cepat kembali ke pemanggilnya, gunakan “Step Out” (Shift + F8), yang akan menyelesaikan eksekusi sisa fungsi lalu menghentikan debug tepat setelah fungsi itu dipanggil. Dengan kombinasi breakpoint, inspeksi variabel, dan kontrol alur eksekusi melalui Step Into, Step Over, dan Step Out, debugger membantu Anda memahami perilaku aplikasi dan menemukan kesalahan secara lebih efisien dibandingkan hanya mengandalkan logging.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



Gambar 4. 4. Contoh Penggunaan debugger

Jawab:

Application class di Android adalah komponen tingkat atas yang mewakili proses aplikasi dan dibuat sebelum komponen lain (Activity, Service, dll.) diinisialisasi; dengan mewarisi Application dan mendaftarkannya di AndroidManifest.xml, Anda dapat menyediakan titik pusat untuk inisialisasi dependensi (misalnya DI framework seperti Hilt atau Dagger), konfigurasi global (seperti logging, analytics, atau error reporting), serta menyimpan state atau objek singleton yang perlu bertahan selama lifecycle aplikasi. Karena instance Application hanya dibuat sekali sepanjang masa hidup proses, ia ideal untuk menyimpan resource bersama, context aplikasi yang aman dari memory leak, dan menjalankan kode setup sebelum UI tampil, sehingga membantu menjaga kebersihan arsitektur dengan memisahkan logika inisialisasi global dari Activity atau Fragment.

MODUL 5 : CONNECT TO THE INTERNET

SOAL 1

Soal Praktikum:

1. Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:
    a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
    b. Gunakan KotlinX Serialization sebagai library JSON.
    c. Gunakan library seperti Coil atau Glide untuk image loading.
    d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started
    e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
    f. Gunakan caching strategy pada Room..
    g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

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

A. Source Code

1. AppDatabase

```
1  package com.example.myapi_test
2
3  import android.content.Context
4  import androidx.room.Database
5  import androidx.room.Room
6  import androidx.room.RoomDatabase
7
8  @Database(entities = [CharacterInfoEntity::class], version =
9  1, exportSchema = false)
10 abstract class AppDatabase : RoomDatabase() {
11
12     abstract fun characterDao(): CharacterDao
13
14     companion object {
15         @Volatile
16         private var INSTANCE: AppDatabase? = null
17
18         fun getDatabase(context: Context): AppDatabase {
19             return INSTANCE ?: synchronized(this) {
20                 val instance = Room.databaseBuilder(
21                     context.applicationContext,
22                     AppDatabase::class.java,
23                     "character_database"
24                 ).build()
25                 INSTANCE = instance
26                 instance
27             }
28         }
29     }
30 }
```

Tabel 5. 1. Source Code AppDatabase.kt

2. CacheMapper.kt

```
1   package com.example.myapi_test
2
3   fun CharacterInfo.toCharacterInfoEntity():
4   CharacterInfoEntity {
5       return CharacterInfoEntity(
6           characterId = this.characterId,
7           characterName = this.characterName,
8           characterUrl = this.characterUrl,
9           characterImageUrl = this.characterImageUrl,
10          japaneseVoiceActor = this.japaneseVoiceActor,
11          englishVoiceActor = this.englishVoiceActor,
12          favorites = this.favorites
13      )
14  }
15
16  fun CharacterInfoEntity.toCharacterInfo(): CharacterInfo {
17      return CharacterInfo(
18          characterId = this.characterId,
19          characterName = this.characterName,
20          characterUrl = this.characterUrl,
21          characterImageUrl = this.characterImageUrl,
22          japaneseVoiceActor = this.japaneseVoiceActor,
23          englishVoiceActor = this.englishVoiceActor,
24          favorites = this.favorites
25      )
26  }
```

Tabel 5. 2. Source Code CacheMapper.kt

3. CharacterAdapter.kt

```
1   package com.example.myapi_test
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.view.LayoutInflater
6   import android.view.ViewGroup
7   import androidx.recyclerview.widget.RecyclerView
    import com.bumptech.glide.Glide
8   import com.example.myapi_test.databinding.ItemListBinding
9   import java.text.NumberFormat
    import java.util.Locale
10
    class CharacterAdapter(
11      private var characters: List<CharacterInfo>,
12      private val onDetailClick: (CharacterInfo) -> Unit
13  ) :
14  RecyclerView.Adapter<CharacterAdapter.CharacterViewHolder>()
15  {
16
17      inner class CharacterViewHolder(val binding:
18  ItemListBinding) :
19          RecyclerView.ViewHolder(binding.root) {
20
21          fun bind(character: CharacterInfo) {
22              binding.apply {
23                  textViewCharacterName.text =
24  character.characterName
25                  val formattedFavorites =
26  NumberFormat.getNumberInstance(Locale.US).format(character.f
27  avorites)
28                  textViewFavorites.text = formattedFavorites
29
30                  val vaJapaneseText = "JP:
31  ${character.japaneseVoiceActor ?: "N/A"}"
32                  val vaEnglishText = "EN:
```

```
33  ${character.englishVoiceActor ?: "N/A"}"
                textViewVoiceActors.text =
34  "$vaJapaneseText\n$vaEnglishText"
35
36                  Glide.with(imageViewCharacter.context)
37                      .load(character.characterImageUrl)
38                      .into(imageViewCharacter)
39
40                  buttonDetail.setOnClickListener {
41                      onDetailClick(character)
42                  }
43
44                  buttonUrl.setOnClickListener {
45                      val intent = Intent(Intent.ACTION_VIEW,
46  Uri.parse(character.characterUrl))
47
    imageViewCharacter.context.startActivity(intent)
48                  }
49              }
50          }
51      }
52
53      override fun onCreateViewHolder(parent: ViewGroup,
54  viewType: Int): CharacterViewHolder {
55          val binding =
56  ItemListBinding.inflate(LayoutInflater.from(parent.context),
57  parent, false)
58          return CharacterViewHolder(binding)
59      }
60
61      override fun getItemCount() = characters.size
62
63      override fun onBindViewHolder(holder:
64  CharacterViewHolder, position: Int) {
65          holder.bind(characters[position])
```

```
66      }
67
68      fun setData(newCharacters: List<CharacterInfo>) {
69          characters = newCharacters
70          notifyDataSetChanged()
71      }
72  }
```

Tabel 5. 3. Source Code CharacterAdapter.kt

4. CharacterDao.kt

```
1   package com.example.myapi_test
2
3   import androidx.room.Dao
4   import androidx.room.Insert
5   import androidx.room.OnConflictStrategy
6   import androidx.room.Query
7   import kotlinx.coroutines.flow.Flow
8
9   @Dao
10  interface CharacterDao {
11
12      @Query("SELECT * FROM characters")
13      fun getCharacters(): Flow<List<CharacterInfoEntity>>
14
15      @Insert(onConflict = OnConflictStrategy.REPLACE)
16      suspend fun insertAll(characters:
17  List<CharacterInfoEntity>)
18
19      @Query("DELETE FROM characters")
20      suspend fun clearAll()
21  }
```

Tabel 5. 4. Source Code CharacterDao.kt

5. CharacterInfoEntity.kt

```
1   package com.example.myapi_test
2
3   import androidx.room.Entity
4   import androidx.room.PrimaryKey
5
6   @Entity(tableName = "characters")
7   data class CharacterInfoEntity(
8       @PrimaryKey
9       val characterId: Int,
10      val characterName: String,
11      val characterUrl: String,
12      val characterImageUrl: String,
13      val japaneseVoiceActor: String?,
14      val englishVoiceActor: String?,
15      val favorites: Int
16  )
```

Tabel 5. 5. Source Code CharacterInfoEntity.kt

6. CharacterMapper.kt

```
1   package com.example.myapi_test
2
3   fun mapToCharacterInfoList(characterDataList:
4   List<CharacterListItem>?): List<CharacterInfo> {
5       return characterDataList?.map { data ->
6           CharacterInfo(
7               characterId = data.character.malId,
8               characterName = data.character.name ?: "Name not
9   found",
10              characterUrl = data.character.url,
11              characterImageUrl =
12  data.character.images?.jpg?.imageUrl ?: "",
13
14              japaneseVoiceActor = data.voices?.find
15  { it.language.equals("Japanese", ignoreCase =
```

```
16  true) }?.person?.name,
17              englishVoiceActor = data.voices?.find
18  { it.language.equals("English", ignoreCase =
19  true) }?.person?.name,
20              favorites = data.favorites
21          )
22      } ?: emptyList()
23  }
```

Tabel 5. 6. Source Code CharacterMapper.kt

7. CharacterModels.kt

```
1   package com.example.myapi_test
2   import android.os.Parcelable
3   import kotlinx.parcelize.Parcelize
4
5   import com.google.gson.annotations.SerializedName
6
7   @Parcelize
8   data class CharacterInfo(
9       val characterId: Int,
10      val characterName: String,
11      val characterUrl: String,
12      val characterImageUrl: String,
13      val japaneseVoiceActor: String?,
14      val englishVoiceActor: String?,
15      val favorites: Int
16  ) : Parcelable
17
18  data class AnimeCharactersResponse(
19      @SerializedName("data")
20      val data: List<CharacterListItem>
21  )
22
23  data class Character(
```

```
24      @SerializedName("mal_id")
25      val malId: Int,
26
27      @SerializedName("url")
28      val url: String,
29
30      @SerializedName("images")
31      val images: Images?,
32
33      @SerializedName("name")
34      val name: String?
35  )
36
37  data class CharacterListItem(
38      @SerializedName("character")
39      val character: Character,
40
41      @SerializedName("role")
42      val role: String,
43
44      @SerializedName("favorites")
45      val favorites: Int,
46
47      @SerializedName("voice_actors")
48      val voices: List<Voice>?
49  )
50
51  data class CharacterDetailResponse(
52      @SerializedName("data")
53      val data: CharacterDetails
54  )
55
56  data class CharacterDetails(
57      @SerializedName("mal_id")
58      val malId: Int,
```

```
59
60      @SerializedName("url")
61      val url: String,
62
63      @SerializedName("images")
64      val images: Images,
65
66      @SerializedName("name")
67      val name: String,
68
69      @SerializedName("name_kanji")
70      val nameKanji: String?,
71
72      @SerializedName("favorites")
73      val favorites: Int,
74
75      @SerializedName("about")
76      val about: String?,
77
78      @SerializedName("voices")
79      val voices: List<Voice>
80  )
81
82  data class Voice(
83      @SerializedName("person")
84      val person: Person,
85      @SerializedName("language")
86      val language: String
87  )
88
89  data class Person(
90      @SerializedName("mal_id")
91      val malId: Int,
92      @SerializedName("url")
93      val url: String,
```

```
94      @SerializedName("images")
95      val images: PersonImages,
96      @SerializedName("name")
97      val name: String
98  )
99
    data class Images(
        @SerializedName("jpg")
        val jpg: ImageType
    )

    data class ImageType(
        @SerializedName("image_url")
        val imageUrl: String
    )

    data class PersonImages(
        @SerializedName("jpg")
        val jpg: ImageType
    )
```

Tabel 5. 7. Source Code CharacterModels.kt

8. CharacterRepository.kt

```
1   package com.example.myapi_test
2
3   import kotlinx.coroutines.flow.Flow
4   import kotlinx.coroutines.flow.map
5
6   class CharacterRepository(
7       private val apiService: JikanApiService,
8       private val characterDao: CharacterDao
9   ) {
10
11      fun getAnimeCharacters(): Flow<List<CharacterInfo>> {
```

```
        return characterDao.getCharacters().map { entities ->
            entities.map { it.toCharacterInfo() }
        }
    }

    suspend fun refreshCharacters(animeId: Int) {
        val response = apiService.getAnimeCharacters(animeId)
        val characterInfoList = mapToCharacterInfoList(response.data)

        characterDao.clearAll()
        characterDao.insertAll(characterInfoList.map { it.toCharacterInfoEntity() })
    }

    suspend fun getCharacterDetails(characterId: Int): CharacterDetailResponse {
        return apiService.getCharacterDetails(characterId)
    }
}

```

Tabel 5. 8. Source Code CharacterRespository.kt

9. CharacterViewModelFactory.kt

```
package com.example.myapi_test

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class CharacterViewModelFactory(private val repository: CharacterRepository) : ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass:
```

```
8   Class<T>): T {
9           return when {
10
11  modelClass.isAssignableFrom(HomeViewModel::class.java) -> {
12                  HomeViewModel(repository) as T
13              }
14
15  modelClass.isAssignableFrom(DetailViewModel::class.java) ->
16  {
17                  DetailViewModel(repository) as T
18              }
19              else -> throw IllegalArgumentException("Unknown
20  ViewModel class: ${modelClass.name}")
21          }
22      }
23  }
```

Tabel 5. 9. Source Code CharacterViewModelFactory.kt

10. DetailFragment.kt

```
1   package com.example.myapi_test
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.util.Log
7   import android.view.LayoutInflater
8   import android.view.View
9   import android.view.ViewGroup
10  import androidx.fragment.app.Fragment
11  import androidx.lifecycle.ViewModelProvider
12  import androidx.navigation.fragment.navArgs
13  import com.bumptech.glide.Glide
14  import
15  com.example.myapi_test.databinding.FragmentDetailBinding
```

```
16 import java.text.NumberFormat
17 import java.util.Locale
18
19 class DetailFragment : Fragment() {
20
21     private var _binding: FragmentDetailBinding? = null
22     private val binding get() = _binding!!
23     private val args: DetailFragmentArgs by navArgs()
24
25     private lateinit var viewModel: DetailViewModel
26
27     override fun onCreateView(
28         inflater: LayoutInflater, container: ViewGroup?,
29         savedInstanceState: Bundle?
30     ): View {
31         _binding = FragmentDetailBinding.inflate(inflater,
32 container, false)
33         return binding.root
34     }
35
36     override fun onViewCreated(view: View,
37 savedInstanceState: Bundle?) {
38         super.onViewCreated(view, savedInstanceState)
39
40         val characterDao =
41 AppDatabase.getDatabase(requireContext()).characterDao()
42         val repository =
43 CharacterRepository(RetrofitInstance.api, characterDao)
44         val viewModelFactory =
45 CharacterViewModelFactory(repository)
46
47         viewModel = ViewModelProvider(this,
48 viewModelFactory)[DetailViewModel::class.java]
49
50         bindInitialData(args.character)
```

```
51
52          setupObservers()
53
54  viewModel.fetchCharacterDetails(args.character.characterId)
55      }
56
57      private fun bindInitialData(character: CharacterInfo) {
58          binding.apply {
59              textViewNameDetail.text =
60  character.characterName
61              val formattedFavorites =
62  NumberFormat.getNumberInstance(Locale.US)
63                  .format(character.favorites)
64              textViewFavoritesDetail.text =
65  "$formattedFavorites Favorites"
66              textViewVoiceActorsList.text = "Loading voice
67  actors..."
68
69              Glide.with(this@DetailFragment)
70                  .load(character.characterImageUrl)
71                  .into(imageViewCharacterDetail)
72
73              buttonViewProfileDetail.setOnClickListener {
74                  val intent = Intent(Intent.ACTION_VIEW,
75  Uri.parse(character.characterUrl))
76                  context?.startActivity(intent)
77              }
78          }
79      }
80
81      private fun setupObservers() {
82
83  viewModel.characterDetails.observe(viewLifecycleOwner)
84  { details ->
85              details?.let { bindFullData(it) }
```

```
86          }
87
88          viewModel.error.observe(viewLifecycleOwner)
89  { errorMessage ->
90              errorMessage?.let {
91                  binding.textViewVoiceActorsList.text = it
92                  Log.e("DetailFragment", "Error: $it")
93              }
94          }
95      }
96
97      private fun bindFullData(details: CharacterDetails) {
98          binding.apply {
99              textViewNameDetail.text = details.name
                val formattedFavorites =
    NumberFormat.getNumberInstance(Locale.US)
                    .format(details.favorites)
                textViewFavoritesDetail.text =
    "$formattedFavorites Favorites"

                val voiceActorsText =
    details.voices.joinToString("\n") { voice ->
                    "${voice.language}: ${voice.person.name}"
                }

                textViewVoiceActorsList.text =
    voiceActorsText.ifEmpty { "No voice actor information
    available." }
            }
        }

        override fun onDestroyView() {
            super.onDestroyView()
            _binding = null
```

```
    }
}
```

Tabel 5. 10. Source Code DetailFragment.kt

11. DetailViewModel.kt

```
1  package com.example.myapi_test
2
3  import androidx.lifecycle.LiveData
4  import androidx.lifecycle.MutableLiveData
5  import androidx.lifecycle.ViewModel
6  import androidx.lifecycle.viewModelScope
7  import kotlinx.coroutines.launch
8  import java.io.IOException
9
10 class DetailViewModel(private val repository:
11 CharacterRepository) : ViewModel() {
12
13     private val _characterDetails =
14 MutableLiveData<CharacterDetails>()
15     val characterDetails: LiveData<CharacterDetails> get() =
16 _characterDetails
17
18     private val _error = MutableLiveData<String?>()
19     val error: LiveData<String?> get() = _error
20
21     fun fetchCharacterDetails(characterId: Int) {
22         viewModelScope.launch {
23             try {
24                 val response =
25 repository.getCharacterDetails(characterId)
26                 _characterDetails.value = response.data
27                 _error.value = null //
28             } catch (e: IOException) {
29                 _error.value = "Failed to load details due
```

```
30  to a network error."
31              } catch (e: Exception) {
32                  _error.value = "An unexpected error occurred
33  while fetching details."
34              }
35          }
36      }
37  }
```

Tabel 5. 11. Source Code DetailViewModel.kt

12. HomeFragment.kt

```
1   package com.example.myapi_test
2
3   import android.os.Bundle
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import android.widget.Toast
9   import androidx.fragment.app.Fragment
10  import androidx.lifecycle.ViewModelProvider
11  import androidx.navigation.findNavController
12  import androidx.recyclerview.widget.LinearLayoutManager
13  import
14  com.example.myapi_test.databinding.FragmentHomeBinding
15
16  class HomeFragment : Fragment() {
17
18      private var _binding: FragmentHomeBinding? = null
19      private val binding get() = _binding!!
20
21      private lateinit var characterAdapter: CharacterAdapter
22      private lateinit var viewModel: HomeViewModel
23
24      override fun onCreateView(
```

```
25          inflater: LayoutInflater, container: ViewGroup?,
26          savedInstanceState: Bundle?
27      ): View {
28          _binding = FragmentHomeBinding.inflate(inflater,
29  container, false)
30          return binding.root
31      }
32
33      override fun onViewCreated(view: View,
34  savedInstanceState: Bundle?) {
35          super.onViewCreated(view, savedInstanceState)
36
37          val characterDao =
38  AppDatabase.getDatabase(requireContext()).characterDao()
39
40          val repository =
41  CharacterRepository(RetrofitInstance.api, characterDao)
42          val viewModelFactory =
43  CharacterViewModelFactory(repository)
44
45          viewModel = ViewModelProvider(this,
46  viewModelFactory)[HomeViewModel::class.java]
47
48          setupRecyclerView()
49          setupObservers()
50      }
51
52      private fun setupRecyclerView() {
53          characterAdapter = CharacterAdapter(emptyList())
54  { character ->
55              val action =
56  HomeFragmentDirections.actionHomeFragmentToDetailFragment(ch
57  aracter)
58
59  requireActivity().findNavController(R.id.nav_host_fragment).
```

```
navigate(action)
        }
        binding.recyclerView.apply {
            layoutManager = LinearLayoutManager(context)
            adapter = characterAdapter
        }
    }

    private fun setupObservers() {
        viewModel.characters.observe(viewLifecycleOwner)
{ characters ->
            if (characters.isNullOrEmpty()) {
                binding.textViewEmptyCache.visibility =
View.VISIBLE
                binding.recyclerView.visibility = View.GONE
            } else {
                binding.textViewEmptyCache.visibility =
View.GONE
                binding.recyclerView.visibility =
View.VISIBLE
                characterAdapter.setData(characters)
            }
        }

        viewModel.isLoading.observe(viewLifecycleOwner)
{ isLoading ->
            binding.progressBar.visibility = if (isLoading)
View.VISIBLE else View.GONE
        }

        viewModel.error.observe(viewLifecycleOwner)
{ errorMessage ->
            errorMessage?.let {
                Toast.makeText(context, it,
Toast.LENGTH_LONG).show()
```

```
                    Log.e("HomeFragment", "Error: $it")
                }
            }
        }

        override fun onDestroyView() {
            super.onDestroyView()
            _binding = null
        }
    }
```

Tabel 5. 12. Source Code HomeFragment.kt

13. HomeViewModel.kt

```
package com.example.myapi_test

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.asLiveData
import androidx.lifecycle.viewModelScope
import kotlinx.coroutines.launch
import java.io.IOException

class HomeViewModel(private val repository:
CharacterRepository) : ViewModel() {

    val characters: LiveData<List<CharacterInfo>> =
repository.getAnimeCharacters().asLiveData()

    private val _isLoading = MutableLiveData<Boolean>()
    val isLoading: LiveData<Boolean> get() = _isLoading

    private val _error = MutableLiveData<String?>()
    val error: LiveData<String?> get() = _error
```

```
22
23      init {
24          refreshCharacterData()
25      }
26
27      fun refreshCharacterData() {
28          viewModelScope.launch {
29              _isLoading.value = true
30              try {
31                  repository.refreshCharacters(52991)
32                  _error.value = null
33              } catch (e: IOException) {
34                  _error.value = "Network error. Displaying
35  cached data."
36              } catch (e: Exception) {
37                  _error.value = "An unexpected error occurred
38  during refresh."
39              } finally {
40                  _isLoading.value = false
41              }
42          }
43      }
44  }
```

Tabel 5. 13. Source Code HomeViewModel.kt

14. JikanApiService.kt

```
1   package com.example.myapi_test
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.navigation.NavController
6   import androidx.navigation.fragment.NavHostFragment
7   import
8   androidx.navigation.ui.setupActionBarWithNavController
9   import
```

```
10  androidx.core.splashscreen.SplashScreen.Companion.installSp
11  lashScreen
12  import
    com.example.myapi_test.databinding.ActivityMainBinding
13
14  class MainActivity : AppCompatActivity() {
15      private lateinit var navController: NavController
16      private lateinit var binding: ActivityMainBinding
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          installSplashScreen()
21          binding =
22  ActivityMainBinding.inflate(layoutInflater)
23          setContentView(binding.root)
24
25          setSupportActionBar(binding.toolbar)

26          val navHostFragment = supportFragmentManager
                .findFragmentById(R.id.nav_host_fragment) as
27  NavHostFragment
28          navController = navHostFragment.navController
29
30          setupActionBarWithNavController(navController)
31      }
32
33      override fun onSupportNavigateUp(): Boolean {
34          return navController.navigateUp() ||
35  super.onSupportNavigateUp()
36      }
37  }
```

Tabel 5. 14. Source Code JikanApiService.kt

15. MainActivity.kt

```
1  package com.example.myapi_test
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import androidx.navigation.NavController
6  import androidx.navigation.fragment.NavHostFragment
7  import
8  androidx.navigation.ui.setupActionBarWithNavController
9  import
10 androidx.core.splashscreen.SplashScreen.Companion.installSp
11 lashScreen
12 import
13 com.example.myapi_test.databinding.ActivityMainBinding
14
15 class MainActivity : AppCompatActivity() {
16     private lateinit var navController: NavController
17     private lateinit var binding: ActivityMainBinding
18
19     override fun onCreate(savedInstanceState: Bundle?) {
20         super.onCreate(savedInstanceState)
21         installSplashScreen()
22         binding =
23 ActivityMainBinding.inflate(layoutInflater)
24         setContentView(binding.root)
25
26         setSupportActionBar(binding.toolbar)
27
28         val navHostFragment = supportFragmentManager
29             .findFragmentById(R.id.nav_host_fragment) as
30 NavHostFragment
31         navController = navHostFragment.navController
32
33         setupActionBarWithNavController(navController)
34     }
35
```

```
36      override fun onSupportNavigateUp(): Boolean {
37          return navController.navigateUp() ||
38  super.onSupportNavigateUp()
39      }
40  }
```

Tabel 5. 15. Source Code MainActivity.kt

16. Retrofit Instance.kt

```
1   package com.example.myapi_test
2
3   import retrofit2.Retrofit
4   import retrofit2.converter.gson.GsonConverterFactory
5
6   object RetrofitInstance {
7       private const val BASE_URL =
8   "https://api.jikan.moe/v4/"
9
10      private val retrofit by lazy {
11          Retrofit.Builder()
12              .baseUrl(BASE_URL)
13              .addConverterFactory(GsonConverterFactory.creat
14  e())
15              .build()
16      }
17
18      val api: JikanApiService by lazy {
19          retrofit.create(JikanApiService::class.java)
20      }
21  }
```

Tabel 5. 16. Source Code RetrofitInstance.kt

17. activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3
4   xmlns:android="http://schemas.android.com/apk/res/android"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent"
9       tools:context=".MainActivity"
10
11      android:background="@color/backgroundColor">
12
13      <com.google.android.material.appbar.MaterialToolbar
14          android:id="@+id/toolbar"
            android:layout_width="match_parent"
15          android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
16          app:titleTextColor="?attr/colorOnPrimary"
17          app:layout_constraintTop_toTopOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintEnd_toEndOf="parent" />

20      <androidx.fragment.app.FragmentContainerView
21          android:id="@+id/nav_host_fragment"
22
23  android:name="androidx.navigation.fragment.NavHostFragment"
24          android:layout_width="0dp"
25          android:layout_height="0dp"
26          app:layout_constraintLeft_toLeftOf="parent"
27          app:layout_constraintRight_toRightOf="parent"
28          app:layout_constraintTop_toBottomOf="@id/toolbar"
29          app:layout_constraintBottom_toBottomOf="parent"
30          app:defaultNavHost="true"
31          app:navGraph="@navigation/nav_graph" />
32
```

```
33
</androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 5. 17. Source Code activity_main.xml

18. fragment_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:fillViewport="true"
8      android:background="@color/backgroundColor"
9      tools:context=".DetailFragment">
10
11     <LinearLayout
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:gravity="center_horizontal"
           android:orientation="vertical"
15         android:padding="16dp">

16         <ImageView
17             android:id="@+id/imageViewCharacterDetail"
18             android:layout_width="200dp"
19             android:layout_height="300dp"

20 android:contentDescription="@string/character_image_descrip
21 tion"
22             android:scaleType="centerCrop"
23             tools:src="@tools:sample/avatars" />
24
25         <TextView
26             android:id="@+id/textViewNameDetail"
```

```
27              android:layout_width="wrap_content"
28              android:layout_height="wrap_content"
29              android:layout_marginTop="16dp"
30              android:textSize="24sp"
31              android:textStyle="bold"
32              android:textColor="@color/textColorPrimary"
33              tools:text="Character Name" />
34
35          <TextView
36              android:id="@+id/textViewFavoritesDetail"
37              android:layout_width="wrap_content"
38              android:layout_height="wrap_content"
39              android:layout_marginTop="8dp"
40              android:textSize="16sp"
41              android:textColor="@color/textColorSecondary"
42              tools:text="123,456 Favorites" />
43
44          <TextView
45              android:id="@+id/textViewVoiceActorsList"
46              android:layout_width="wrap_content"
47  android:layout_height="wrap_content"
48              android:layout_marginTop="16dp"
49              android:textAlignment="center"
50              android:textSize="14sp"
51              android:textColor="@color/textColorSecondary"
52              tools:text="Japanese: VA Name\nEnglish: VA
53  Name" />
54
55          <Button
56              android:id="@+id/buttonViewProfileDetail"
57              android:layout_width="wrap_content"
58              android:layout_height="wrap_content"
59              android:layout_marginTop="24dp"
60              android:text="@string/view_profile_button" />
61
```

```
    </LinearLayout>
</ScrollView>
```

Tabel 5. 18. Source Code fragment_detail.xml

19. fragment_home.xml

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".HomeFragment"

    android:background="@color/backgroundColor">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />

    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27          app:layout_constraintTop_toTopOf="parent"
28          tools:visibility="visible" />
29
30      <TextView
31          android:id="@+id/text_view_empty_cache"
32          android:layout_width="wrap_content"
33          android:layout_height="wrap_content"
34          android:text="No cached data available. Please
35  check your network."
36          android:visibility="gone"
37
38          android:textColor="@color/textColorSecondary"
39
40          app:layout_constraintTop_toTopOf="parent"
41          app:layout_constraintBottom_toBottomOf="parent"
42          app:layout_constraintStart_toStartOf="parent"
43          app:layout_constraintEnd_toEndOf="parent" />
44
45  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 5. 19. Source Code fragment_home.xml

20. item_list.xml

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


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

```
10      app:cardBackgroundColor="@color/cardBackgroundColor">
11
12      <androidx.constraintlayout.widget.ConstraintLayout
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:padding="16dp">
16          <ImageView
17              android:id="@+id/imageViewCharacter"
18              android:layout_width="100dp"
                android:layout_height="150dp"
19              android:scaleType="centerCrop"
                app:layout_constraintStart_toStartOf="parent"
20              app:layout_constraintTop_toTopOf="parent"
21              tools:src="@tools:sample/avatars"
22
23  android:contentDescription="@string/character_image_descrip
24  tion" />
25
26          <TextView
27              android:id="@+id/textViewCharacterName"
28              android:layout_width="0dp"
                android:layout_height="wrap_content"
29              android:layout_marginStart="16dp"
30              android:textSize="20sp"
31              android:textStyle="bold"
32              android:textColor="@color/textColorPrimary"
33              app:layout_constraintEnd_toEndOf="parent"
34
    app:layout_constraintStart_toEndOf="@id/imageViewCharacter"
35
36  app:layout_constraintTop_toTopOf="@id/imageViewCharacter"
37              tools:text="Character Name" />
38
39          <ImageView
```

```
            android:id="@+id/iconFavorites"
            android:layout_width="16dp"
            android:layout_height="16dp"
            android:layout_marginTop="12dp"
            android:src="@drawable/ic_star_gold"
            app:tint="@color/starColor"

app:layout_constraintStart_toStartOf="@id/textViewCharacterName"

app:layout_constraintTop_toBottomOf="@id/textViewCharacterName"

app:layout_constraintBottom_toBottomOf="@id/textViewFavorites"

android:contentDescription="@string/favorites_icon_description" />

        <TextView
            android:id="@+id/textViewFavorites"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="6dp"
            android:textSize="14sp"
            android:textStyle="bold"
            android:textColor="@color/textColorSecondary"

app:layout_constraintTop_toTopOf="@id/iconFavorites"

app:layout_constraintStart_toEndOf="@id/iconFavorites"
            tools:text="98,425" />

        <TextView
            android:id="@+id/textViewVoiceActors"
```

```
68              android:layout_width="0dp"
69              android:layout_height="wrap_content"
70              android:layout_marginTop="12dp"
71              android:textSize="14sp"
72              android:textColor="@color/textColorSecondary"
73
74  app:layout_constraintTop_toBottomOf="@id/textViewFavorites"
75
76  app:layout_constraintStart_toStartOf="@id/textViewCharacter
77  Name"
78              app:layout_constraintEnd_toEndOf="parent"
79              tools:text="JP: Voice Actor\nEN: Voice Actor"
80  />
81
82          <Button
83              android:id="@+id/buttonDetail"
84              android:layout_width="wrap_content"
85              android:layout_height="wrap_content"
86              android:layout_marginTop="16dp"
87              android:text="@string/details_button"
88              app:layout_constraintEnd_toEndOf="parent"
89
90  app:layout_constraintTop_toBottomOf="@id/imageViewCharacter
91  "
92
93  app:layout_constraintBottom_toBottomOf="parent"/>
94
95          <Button
96              android:id="@+id/buttonUrl"
97
98  style="@style/Widget.MaterialComponents.Button.TextButton"
99              android:layout_width="wrap_content"
100             android:layout_height="wrap_content"
101             android:layout_marginEnd="8dp"
102             android:text="@string/view_profile_button"
```

```
103
104 app:layout_constraintBaseline_toBaselineOf="@id/buttonDetai
105 l"
106
107 app:layout_constraintEnd_toStartOf="@id/buttonDetail" />
108
109     </androidx.constraintlayout.widget.ConstraintLayout>
110 </androidx.cardview.widget.CardView>
```

Tabel 5. 20. Source Code item_list.xml

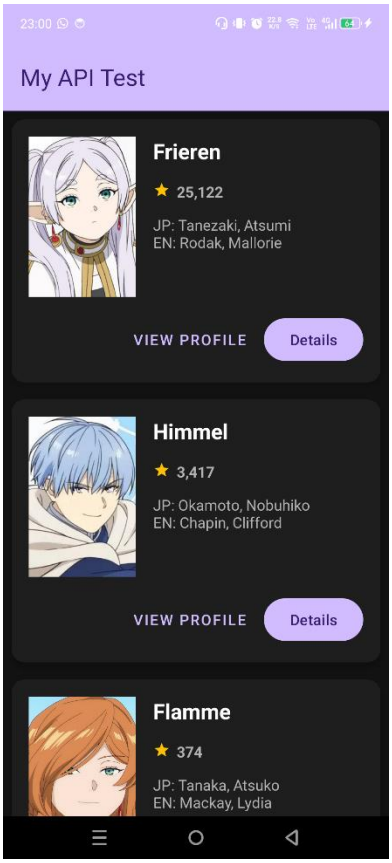
21. nav_graph.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <navigation
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:id="@+id/nav_graph"
7      app:startDestination="@id/homeFragment">
8
9      <fragment
10         android:id="@+id/homeFragment"
11         android:name="com.example.myapi_test.HomeFragment"
12         tools:layout="@layout/fragment_home">
13
14         <action
15
16 android:id="@+id/action_homeFragment_to_detailFragment"
17             app:destination="@id/detailFragment" />
18
19     </fragment>
20
21     <fragment
22         android:id="@+id/detailFragment"
23
24 android:name="com.example.myapi_test.DetailFragment"
```

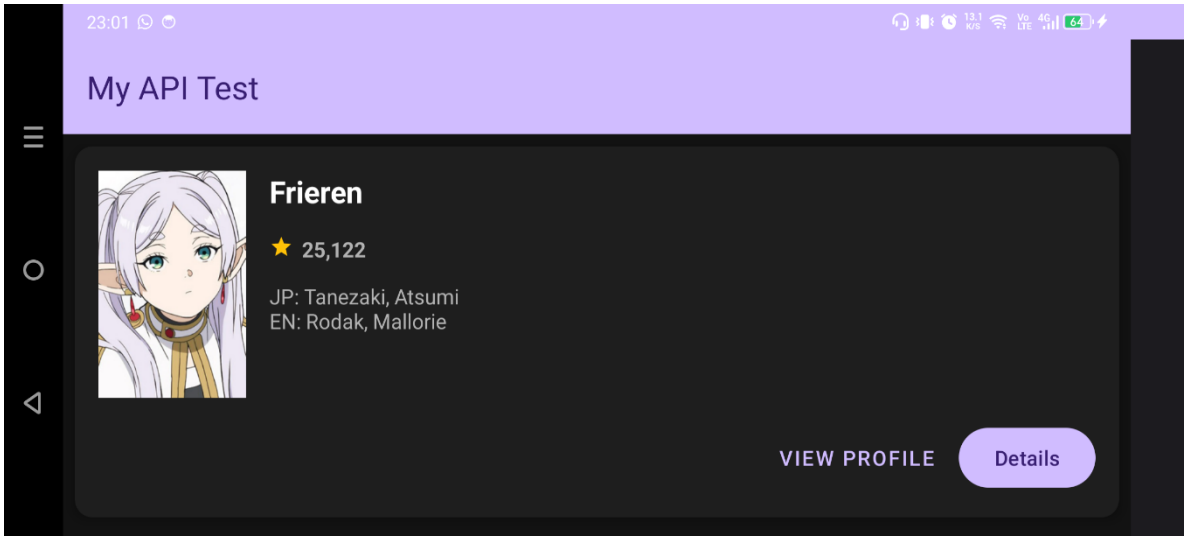
```
25          tools:layout="@layout/fragment_detail">
26
27          <argument
28              android:name="character"
29
30  app:argType="com.example.myapi_test.CharacterInfo" />
31      </fragment>
32
33  </navigation>
```

Tabel 5. 21. Source Code nav_graph.xml

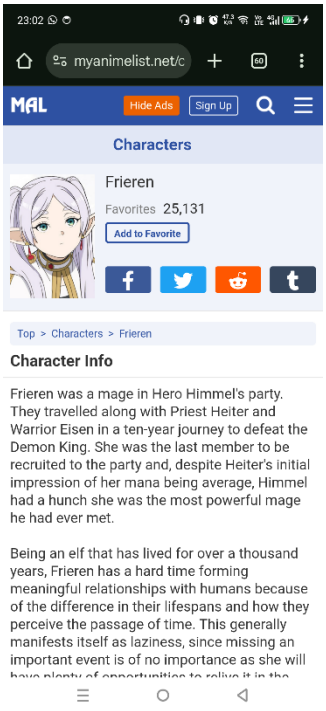
B. Output Program



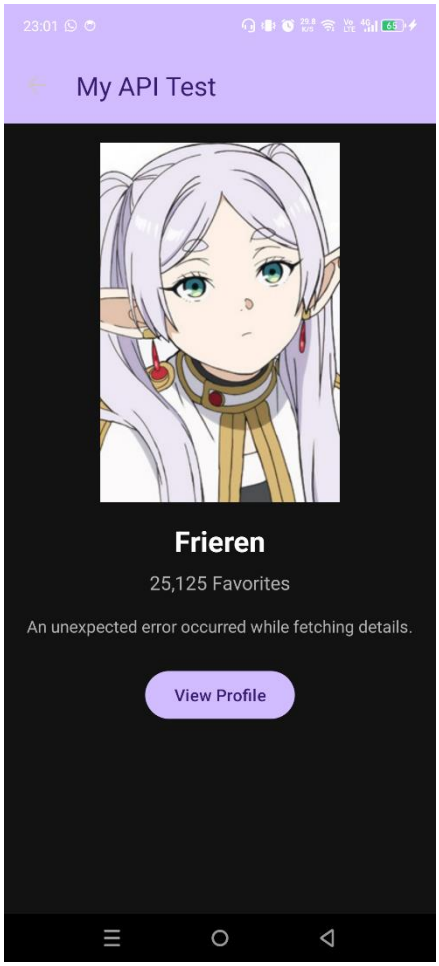
Gambar 5. 1. Screenshot Hasil Jawaban Soal 1



Gambar 5. 2. Screenshot Hasil Jawaban Soal 1



Gambar 5. 3. Screenshot tombol Detail



Gambar 5. 4. Screenshot tombol Info

C. Pembahasan

1. AppDatabase.kt

Digunakan sebagai penghubung utama data base room (local database) dengan cara menyediakan akses ke DAO

2. CacheMapper.kt

Digunakan untuk konversi data misal dari objek CharacterInfo akan di ubah menjadi CharacterInfoEntity dengan tujuan agar bisa di simpan di room.

3. CharacterAdapter.kt

Adapter, dan juga sebagai listener button

4. **CharacterDao.kt**

   DAO buat crud.

5. **CharacterInfoEntity.kt**

   Struktur tabel character untuk database Room

6. **CharacterMapper.kt**

   Konversi data agar bisa dipakai.

7. **CharacterModels.kt**

   Data class utama

8. **CharacterRepository.kt**

   Penguhubung antara remote database, dan local database

9. **CharacterViewModelFactory.kt**

   digunakan untuk membuat instance dari viewmodel yang lain serta untuk memastikan viewmodel memliki depedensi yang di perlukan

10. **DetailFragment.kt**

    Fragment detail yang mengambil data dari navigation, dan view model yang bersangkutan.

11. **DetailViewModel.kt**

    Viewmodel dari detail fragment.

12. **HomeFragment.kt**

HomeFragment, sebagai fragment pertama yang terbuka saat membuka aplikasi

13. HomeViewModel.kt

Viewmodel dari HomeFragment

14. JikanApiService.kt

Wadah untuk membuat request terhadap api

15. MainActivity.kt

Berisi navigasi dan instalasi splash screen

16. RetrofitInstance.kt

Untuk melakukan koneksi terhadap endpoint yang sudah di set.

17. activity_main.xml

Berisi toolbar, dan navhost

18. fragment_detail.xml

Berisi image view, dan text view untuk ui detail fragment

19. fragment_home.xml

Berisi template wadah recylerview yang akan di populate oleh item_list

20. item_list.xml

Card atau ui dari recylerview

21. nav_graph.xml

Navigasi dan data passing

TAUTAN GIT

https://github.com/au290/College-Work/tree/d6f0a2cfdc85584a7e0e036b3b93727b55168a60/Semester-4/Pemrogaman-Mobile