LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Putra Whyra Pratama S. **NIM. 2310817210029**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

JUNI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN WEB II

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

Nama Praktikan : Putra Whyra Pratama S.
NIM : 2310817210029

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

LEMBAR PENGESAHAN 2
DAFTAR ISI 3
DAFTAR GAMBAR 5
DAFTAR TABEL 6
MODUL 1 : Android Basics in Kotlin 9
SOAL 1 9
A. Source Code 11
B. Output Program 14
C. Pembahasan 14
MODUL 2 : Android Layout 16
SOAL 1 16
A. Source Code 17
B. Output Program 22
C. Pembahasan 22
MODUL 3 : Build a Scrollable List 25
SOAL 1 25
A. Source Code 27
B. Output Program 38
C. Pembahasan 38
SOAL 2 45
A. Jawaban 45

MODUL 4 : ViewModel and Debugging 46
SOAL 1 46
A. Source Code 46
B. Output Program 60
C. Pembahasan 61
SOAL 2 73
A. Jawaban 73
MODUL 5 : Connect to the Internet 76
SOAL 1 76
A. Source Code 76
B. Output Program 103
C. Pembahasan 104
Tautan Git 108

DAFTAR GAMBAR

MODUL 1 : Android Basics in Kotlin

Gambar 1. Tampilan Awal Aplikasi 9
Gambar 2. Tampilan Dadu Setelah Di Roll 10
Gambar 3. Tampilan Roll Dadu Double 11
Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1 14

MODUL 2 : Android Layout

Gambar 5. Tampilan Awal Aplikasi 16
Gambar 6. Tampilan Aplikasi Setelah Dijalankan 17
Gambar 7. Screenshot Hasil Jawaban Soal 1 Modul 2 22

MODUL 3 : Build a Scrollable List

Gambar 8. Contoh UI List 26
Gambar 9. Contoh UI Detail 27
Gambar 10. Screenshot Hasil Jawaban Soal 1 Modul 3 38

MODUL 4 : ViewModel and Debugging

Gambar 11. Screenshot Hasil Jawaban Soal 1 Modul 4 60
Gambar 12. Screenshot Log Saat Data Item Masuk Ke Dalam List Modul 4 60
Gambar 13. Screenshot Log Saat Tombol Detail Dan Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Modul 4 60
Gambar 14. Screenshot Log Tombol Explicit Intent Ditekan Modul 4 61

MODUL 5 : Connect to the Internet

Gambar 15. Screenshot Hasil Jawaban Soal 1 Modul 5 103
Gambar 16. Screenshot Log Saat Data Item Masuk Ke Dalam List Modul 5 103
Gambar 17. Screenshot Log Saat Tombol Detail Dan Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Modul 5 103
Gambar 18. Screenshot Log Tombol Explicit Intent Ditekan Modul 5 103

DAFTAR TABEL

MODUL 1 : Android Basics in Kotlin

Tabel 1. Source Code MainActivity.kt 11

MODUL 2 : Android Layout

Tabel 2. Source Code MainActivity.kt 17
Tabel 3. Source Code activity_main.xml 19
Tabel 4. Source Code themes.xml 21

MODUL 3 : Build a Scrollable List

Tabel 5. Source Code MainActivity.kt 27
Tabel 6. Source Code Character.kt 28
Tabel 7. Source Code ListCharacterAdapter.kt 28
Tabel 8. Source Code HomeFragment.kt 30
Tabel 9. Source Code DetailFragment.kt 32
Tabel 10. Source Code activity_main.xml 33
Tabel 11. Source Code fragment_home.xml 33
Tabel 12. Source Code fragment_detail.xml 34
Tabel 13. Source Code item_character.xml 35

MODUL 4 : ViewModel and Debugging

Tabel 14. Source Code MainActivity.kt 46
Tabel 15. Source Code Character.kt 47
Tabel 16. Source Code ListCharacterAdapter.kt 48
Tabel 17. Source Code HomeFragment.kt 49
Tabel 18. Source Code DetailFragment.kt 52
Tabel 19. Source Code activity_main.xml 53
Tabel 20. Source Code fragment_home.xml 53
Tabel 21. Source Code fragment_detail.xml 54
Tabel 22. Source Code item_character.xml 55
Tabel 23. Source Code HomeViewModel.kt 58
Tabel 24. Source Code ViewModelFactory.kt 59

MODUL 5 : Connect to the Internet

Tabel 25. Source Code MainActivity.kt 76
Tabel 26. Source Code ArmorAdapter.kt 77
Tabel 27. Source Code ArmorDao.kt 79
Tabel 28. Source Code ArmorDatabase.kt 79
Tabel 29. Source Code ArmorEntity.kt 80
Tabel 30. Source Code Converters.kt 81
Tabel 31. Source Code ArmorModels.kt 81
Tabel 32. Source Code ApiResponse.kt 83
Tabel 33. Source Code ArmorApiService.kt 83
Tabel 34. Source Code AppContainer.kt 83
Tabel 35. Source Code ArmorDetailFragment.kt 84
Tabel 36. Source Code ArmorListFragment.kt 87
Tabel 37. Source Code ArmorViewModel.kt 90
Tabel 38. Source Code ArmorRepository.kt 92
Tabel 39. Source Code ViewModelFactory.kt 93
Tabel 40. Source Code ArmorApplication.kt 94

Tabel 41. Source Code activity_main.xml 94
Tabel 42. Source Code fragment_armor_detail.xml 94
Tabel 43. Source Code fragment_armor_list.xml 98
Tabel 44. Source Code item_armor.xml 99
Tabel 45. Source Code main_nav_graph.xml 102

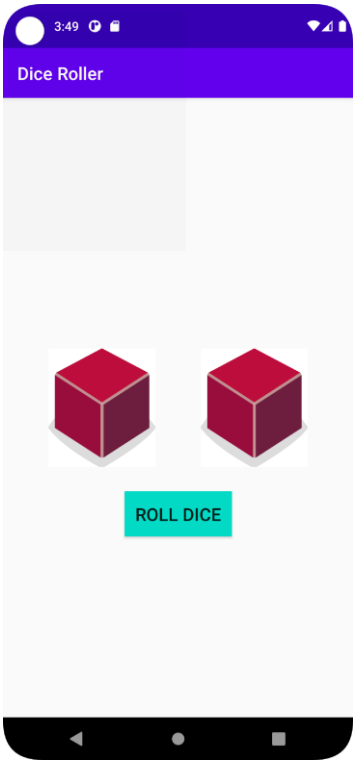
MODUL 1 :

Android Basics in Kotlin

SOAL 1

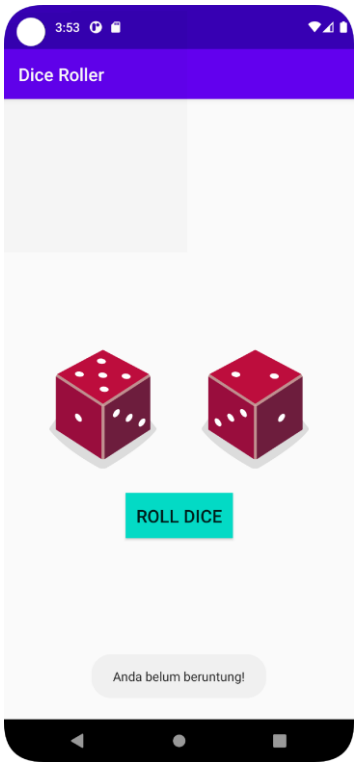
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



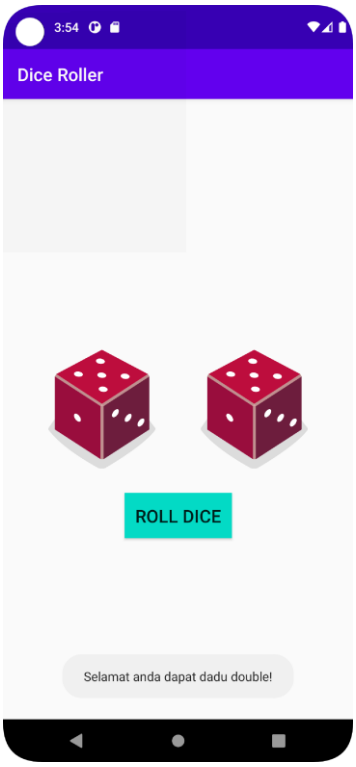
Gambar 1. Tampilan Awal Aplikasi

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



Gambar 2. Tampilan Dadu Setelah Di Roll

3. Apabila user mendapatkan nilai dadu yang sama antara Dadu 1 dan Dadu 2 atau nilai double, maka aplikasi akan menampilkan pesan "Selamat anda dapat dadu double!" seperti dapat dilihat pada Gambar 3.
4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.
5. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export= download



Gambar 3. Tampilan Roll Dadu Double

A. Source Code

1. MainActivity.kt

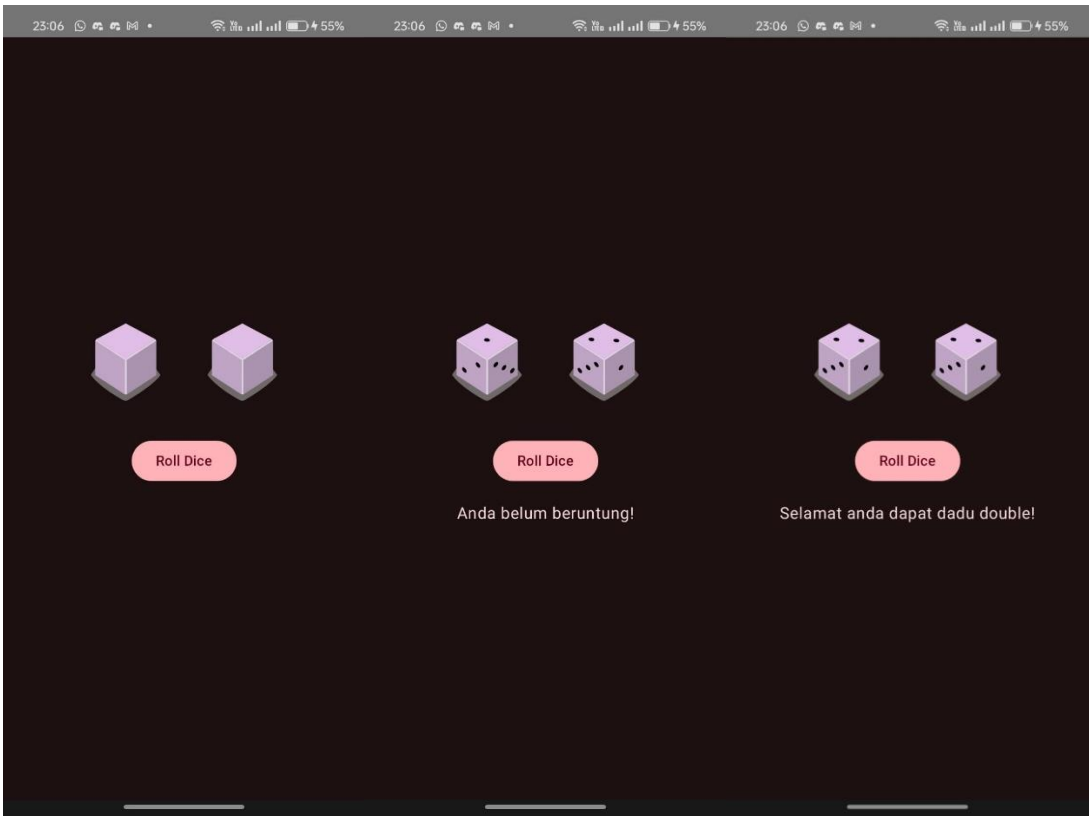
Tabel 1. Source Code MainActivity.kt

```
1   package com.example.diceroller
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.compose.foundation.Image
7   import androidx.compose.foundation.layout.*
8   import androidx.compose.material3.*
9   import androidx.compose.runtime.*
10  import androidx.compose.ui.Alignment
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.res.painterResource
13  import androidx.compose.ui.unit.dp
14  import com.example.diceroller.ui.theme.DiceRollerTheme
15  import kotlin.random.Random
16
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            DiceRollerTheme {
                Surface(modifier =
Modifier.fillMaxSize()) {
                    DiceRollerApp()
                }
            }
        }
    }
}

@Composable
fun DiceRollerApp() {
    var dice1 by remember { mutableStateOf(0) }
    var dice2 by remember { mutableStateOf(0) }
    var message by remember { mutableStateOf("") }

    val getDiceImage = { value: Int ->
        when (value) {
            1 -> R.drawable.dice_1
            2 -> R.drawable.dice_2
            3 -> R.drawable.dice_3
            4 -> R.drawable.dice_4
            5 -> R.drawable.dice_5
            6 -> R.drawable.dice_6
            else -> R.drawable.dice_0
        }
    }

    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment =
Alignment.CenterHorizontally
    ) {
        Row(
            horizontalArrangement =
Arrangement.spacedBy(16.dp)
        ) {
            Image(
                painter = painterResource(id =
getDiceImage(dice1)),
                contentDescription = "Dice 1",
                modifier = Modifier.size(100.dp)
```

```
62              )
63              Image(
64                  painter = painterResource(id =
    getDiceImage(dice2)),
65                  contentDescription = "Dice 2",
66                  modifier = Modifier.size(100.dp)
67              )
68          }
69
70          Spacer(modifier = Modifier.height(24.dp))
71
72          Button(onClick = {
73              dice1 = Random.nextInt(1, 7)
74              dice2 = Random.nextInt(1, 7)
75              message = if (dice1 == dice2) {
76                  "Selamat anda dapat dadu double!"
77              } else {
78                  "Anda belum beruntung!"
79              }
80          }) {
81              Text("Roll Dice")
82          }
83
84          Spacer(modifier = Modifier.height(16.dp))
85
86          Text(text = message)
87      }
88  }
```

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

1. MainActivity.kt:

- Pada baris 1 hingga 10, dilakukan proses impor terhadap pustaka-pustaka yang dibutuhkan, seperti androidx.compose.* untuk antarmuka Jetpack Compose, kotlin.random.Random untuk menghasilkan angka acak, serta tema khusus yang diatur dalam DiceRollerTheme. Hal ini menjadi dasar bagi fungsi visual dan logika dalam aplikasi.
- Masuk ke baris 12 hingga 20, terdapat kelas MainActivity yang merupakan pintu masuk utama aplikasi Android. Di dalam fungsi onCreate, setContent digunakan untuk mendefinisikan isi layar. Komponen Surface di sini digunakan sebagai wadah latar belakang, dan DiceRollerApp() dipanggil untuk menampilkan seluruh konten aplikasi.

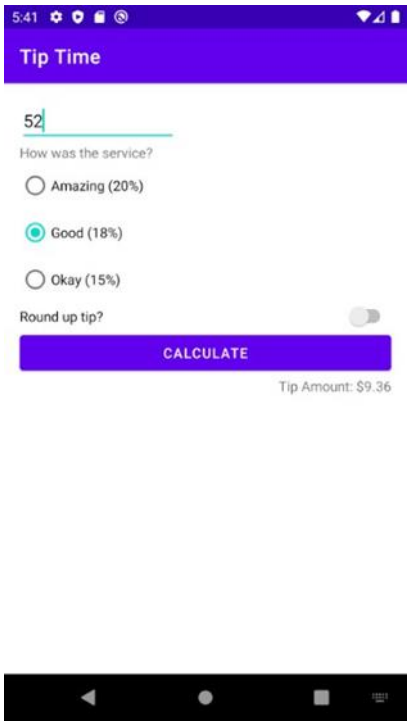
- Selanjutnya, di baris 22 hingga 59, terdapat fungsi DiceRollerApp() yang didekorasi dengan anotasi @Composable, menandakan bahwa ini adalah fungsi UI dalam Compose. Di baris 23–25, tiga buah variabel dice1, dice2, dan message dideklarasikan menggunakan remember, yang memungkinkan state tetap terjaga selama komposisi ulang (recomposition) berlangsung, namun tidak bertahan jika orientasi layar berubah.
- Pada baris 27 hingga 35, terdapat lambda getDiceImage yang digunakan untuk mencocokkan angka hasil lemparan dadu dengan gambar yang sesuai di dalam drawable. Jika angka tidak valid (misalnya 0), maka gambar dice_0 akan digunakan sebagai dadu kosong.
- Komponen UI utama dibangun di baris 37 hingga 59. Column digunakan sebagai layout vertikal utama, dengan Row di dalamnya (baris 42–48) untuk menampilkan dua buah gambar dadu secara horizontal. Kemudian, pada baris 50–55, terdapat Button dengan aksi onClick yang akan mengacak nilai kedua dadu dan memperbarui message sesuai kondisi: jika kedua nilai sama, pesan keberuntungan ditampilkan; jika tidak, pesan kegagalan yang muncul. Di bagian akhir, pada baris 57, teks dari pesan ditampilkan.

MODUL 2 :
Android Layout
SOAL 1

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

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

Gambar 5. Tampilan Awal Aplikasi



Gambar 6. Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1. MainActivity.kt

Tabel 2. Source Code MainActivity.kt

```
1   package com.example.tipcalculator
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.enableEdgeToEdge
7   import androidx.compose.foundation.layout.fillMaxSize
8   import androidx.compose.foundation.layout.padding
9   import androidx.compose.material3.Scaffold
10  import androidx.compose.material3.Text
11  import androidx.compose.runtime.Composable
12  import androidx.compose.ui.Modifier
13  import androidx.compose.ui.tooling.preview.Preview
14  import
    androidx.core.splashscreen.SplashScreen.Companion.instal
    lSplashScreen
```

```
15 import
   com.example.tipcalculator.databinding.ActivityMainBindin
   g
16 import
   com.example.tipcalculator.ui.theme.TipCalculatorTheme
17 import java.text.NumberFormat
18 import java.util.Locale
19 import kotlin.math.ceil
20
21 class MainActivity : ComponentActivity() {
22
23     lateinit var binding: ActivityMainBinding
24
25     override fun onCreate(savedInstanceState: Bundle?) {
26         super.onCreate(savedInstanceState)
27         Thread.sleep(3000)
28         installSplashScreen()
29         binding =
   ActivityMainBinding.inflate(layoutInflater)
30         setContentView(binding.root)
31         binding.calculateButton.setOnClickListener {
   calculateTip() }
32     }
33
34     private fun calculateTip() {
35         val cost =
   binding.costOfService.text.toString().toDouble()
36         val selected =
   binding.tipOptions.checkedRadioButtonId
37         val tipPercentage = when (selected) {
38             R.id.option_twenty_percent -> 0.20
39             R.id.option_eighteen_percent -> 0.18
40             else -> 0.15
41         }
42         var tip = tipPercentage * cost
43         val roundUp = binding.roundTip.isChecked
44         if (roundUp) {
45             tip = ceil(tip)
46         }
47         val currencyTip =
   NumberFormat.getCurrencyInstance(Locale.US).format(tip)
48         binding.tipResult.text =
   getString(R.string.tip_amount, currencyTip)
49     }
50 }
```

2. activity_main.xml

Tabel 3. Source Code activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/linearLayout"
6       android:padding="16dp"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent">
9
10
11      <EditText
12          android:id="@+id/cost_of_service"
13          android:hint="Cost of Service"
14          android:inputType="number"
15          app:layout_constraintTop_toTopOf="parent"
16          app:layout_constraintLeft_toLeftOf="parent"
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content"/>
19
20      <TextView
21          android:id="@+id/service_question"
22          android:layout_width="wrap_content"
23          android:layout_height="wrap_content"
24          app:layout_constraintStart_toStartOf="parent"
25
    app:layout_constraintTop_toBottomOf="@id/cost_of_service
    "
26          android:text="How was the service?"/>
27
28      <RadioGroup
29          android:id="@+id/tip_options"
30          android:orientation="vertical"
31
    android:checkedButton="@id/option_twenty_percent"
32          app:layout_constraintStart_toStartOf="parent"
33
    app:layout_constraintTop_toBottomOf="@id/service_questio
    n"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content">
36
37          <RadioButton
```

```
38              android:id="@+id/option_twenty_percent"
39              android:text="Amazing (20%)"
40              android:layout_width="wrap_content"
41              android:layout_height="wrap_content"/>
42
43          <RadioButton
44              android:id="@+id/option_eighteen_percent"
45              android:text="Good (18%)"
46              android:layout_width="wrap_content"
47              android:layout_height="wrap_content"/>
48
49          <RadioButton
50              android:id="@+id/option_fifteen_percent"
51              android:text="Okay (15%)"
52              android:layout_width="wrap_content"
53              android:layout_height="wrap_content"/>
54
55      </RadioGroup>
56
57      <Switch
58          android:id="@+id/round_tip"
59          android:checked="true"
60
    app:layout_constraintTop_toBottomOf="@id/tip_options"
61
    app:layout_constraintStart_toStartOf="@id/tip_options"
62          app:layout_constraintEnd_toEndOf="parent"
63          android:text="Round up tip?"
64          android:layout_width="0dp"
65          android:layout_height="wrap_content"/>
66
67      <Button
68          android:id="@+id/calculate_button"
69          app:layout_constraintEnd_toEndOf="parent"
70          app:layout_constraintStart_toStartOf="parent"
71
    app:layout_constraintTop_toBottomOf="@id/round_tip"
72          android:text="CALCULATE"
73          android:layout_width="0dp"
74          android:layout_height="wrap_content"/>
75
76      <TextView
77          android:id="@+id/tip_result"
78          android:textSize="20sp"
79          android:textStyle="bold"
80
    app:layout_constraintTop_toBottomOf="@id/calculate_butto
    n"
81          app:layout_constraintEnd_toEndOf="parent"
```

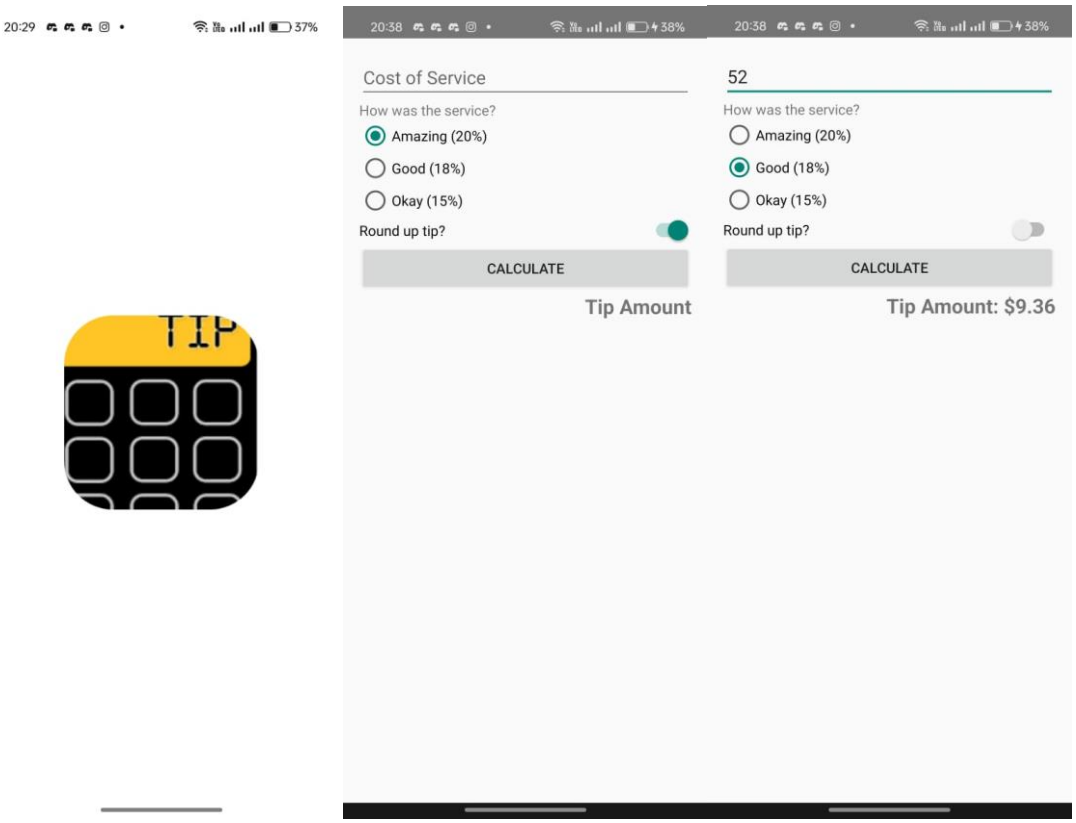
```
82          android:text="Tip Amount"
83          android:layout_width="wrap_content"
84          android:layout_height="wrap_content"/>
85
86  </androidx.constraintlayout.widget.ConstraintLayout>
```

3. themes.xml

Tabel 4. Source Code themes.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <resources>
3
4       <style name="Theme.TipCalculator"
    parent="android:Theme.Material.Light.NoActionBar" />
5       <style name="Theme.App.SplashScreen"
    parent="Theme.SplashScreen">
6           <item
    name="android:windowBackground">@color/white</item>
7           <item
    name="android:windowSplashScreenAnimatedIcon">@drawable/
    tip_calc_splash</item>
8           <item
    name="postSplashScreenTheme">@style/Theme.TipCalculator<
    /item>
9       </style>
10  </resources>
```

B. Output Program



Gambar 7. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

1. MainActivity.kt:

- Pada **baris 1 hingga 19**, dilakukan impor berbagai library yang diperlukan, mulai dari pustaka dasar Android hingga fitur tambahan seperti SplashScreen, View Binding, dan utilitas pemformatan angka dari NumberFormat. Impor ini bertujuan agar program dapat mendukung tampilan modern serta menyediakan perhitungan nilai tip dalam bentuk format mata uang.
- Selanjutnya, **baris 21 sampai 23** mendefinisikan kelas MainActivity yang mewarisi ComponentActivity. Di dalamnya terdapat deklarasi variabel binding dari tipe ActivityMainBinding, yang akan digunakan untuk mengakses elemen layout secara langsung tanpa perlu menggunakan findViewById.

- Bagian utama program terletak pada fungsi onCreate() (**baris 25–32**). Di bagian ini, splash screen diaktifkan menggunakan installSplashScreen() dan diberi jeda 3 detik menggunakan Thread.sleep(3000). Setelah itu, layout di-inflate menggunakan View Binding, dan seluruh isi layout ditampilkan melalui setContentView(). Tombol calculateButton juga diberikan aksi klik (setOnClickListener) yang akan memicu proses perhitungan tip saat ditekan.
- Fungsi calculateTip() yang berada di **baris 34–49** berisi logika utama untuk menghitung jumlah tip. Program pertama-tama mengambil input nilai biaya layanan dari EditText, lalu memeriksa pilihan tip yang dipilih oleh pengguna (20%, 18%, atau 15%). Nilai persentase tersebut kemudian dikalikan dengan biaya layanan untuk menghasilkan besaran tip awal. Jika opsi pembulatan (roundTip) dicentang, maka nilai tip akan dibulatkan ke atas menggunakan fungsi ceil(). Terakhir, nilai tip diformat menjadi mata uang dengan NumberFormat dan ditampilkan ke layar.

2. activity_main.xml:

- Pada **baris 1 hingga 8**, ditentukan struktur dasar layout menggunakan tag ConstraintLayout. Layout ini diberi padding 16dp dan disetel agar ukurannya menyesuaikan dengan ukuran layar (match_parent pada lebar dan tinggi). Selain itu, terdapat deklarasi namespace XML yang diperlukan untuk atribut-atribut khusus Android, seperti tools, app, dan android.
- Bagian input untuk memasukkan nilai layanan ditentukan pada **baris 11 hingga 18** dengan menggunakan EditText. Komponen ini memberikan petunjuk berupa teks “Cost of Service” dan membatasi input hanya berupa angka. EditText ini diatur agar berada di atas layout sebagai elemen pertama yang ditampilkan.
- Selanjutnya, pada **baris 20–26**, terdapat TextView yang menampilkan pertanyaan "How was the service?" sebagai pengantar untuk memilih kualitas layanan. Tepat di bawahnya, **baris 28–55** berisi RadioGroup yang terdiri dari tiga RadioButton, masing-masing untuk memilih persentase tip berdasarkan kualitas layanan: 20% untuk "Amazing", 18% untuk "Good", dan 15% untuk "Okay". Penggunaan RadioGroup ini membuat hanya satu pilihan bisa dipilih dalam satu waktu.
- Di **baris 57–65**, terdapat komponen Switch dengan label “Round up tip?” yang memungkinkan pengguna memilih apakah hasil perhitungan tip ingin dibulatkan ke atas. Switch ini diatur sejajar dengan elemen-elemen sebelumnya agar tata letak tetap rapi dan responsif.
- Tombol untuk melakukan kalkulasi terdapat pada **baris 67–74**, menggunakan Button dengan label “CALCULATE”. Tombol ini menjadi pemicu utama untuk menjalankan fungsi perhitungan tip yang telah didefinisikan di bagian kode Kotlin (activity).
- Terakhir, pada **baris 76–84**, terdapat TextView yang digunakan untuk menampilkan hasil perhitungan tip. Elemen ini diberi ukuran font yang lebih besar (20sp) dan dicetak tebal agar lebih menonjol.

3. **themes.xml:**

- Pada **baris 1 sampai 2**, dituliskan deklarasi XML standar untuk file sumber daya Android, yaitu menggunakan tag <resources>. Ini menandakan bahwa isi file akan berisi definisi sumber daya seperti gaya (style), warna (color), atau dimensi (dimen).
- Selanjutnya, **baris 4** mendefinisikan tema utama aplikasi dengan nama Theme.TipCalculator. Tema ini menggunakan turunan dari android:Theme.Material.Light.NoActionBar, yang berarti aplikasi akan menggunakan tampilan material design dengan latar belakang terang dan tanpa action bar di bagian atas. Pemilihan tema ini membantu membuat tampilan aplikasi lebih bersih dan modern.
- Kemudian, **baris 5 hingga 9** mendefinisikan sebuah style baru dengan nama Theme.App.SplashScreen, yang secara khusus digunakan untuk mengatur tampilan splash screen aplikasi. Style ini menurunkan atribut dari Theme.SplashScreen, yang merupakan tema bawaan Android untuk splash screen.

  Dalam style splash screen tersebut, ada beberapa item penting:
  - **Baris 6** mengatur latar belakang splash screen dengan warna putih (@color/white).
  - **Baris 7** menentukan ikon animasi splash screen, dalam hal ini menggunakan gambar @drawable/tip_calc_splash.
  - **Baris 8** menyatakan bahwa setelah splash screen selesai ditampilkan, tema aplikasi akan diganti kembali ke Theme.TipCalculator sebagai tampilan utama.

MODUL 3 :
Build a Scrollable List
SOAL 1

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

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

   Terdapat 2 button dalam list, dengan fungsi berikut:

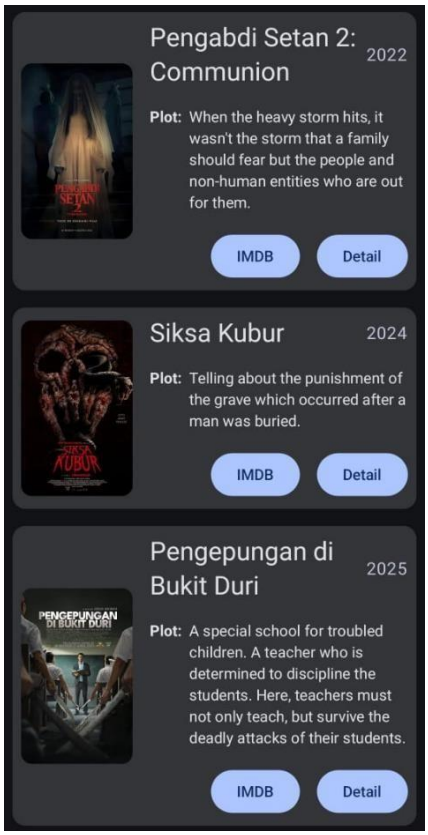
a. Button pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain

b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item

5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi

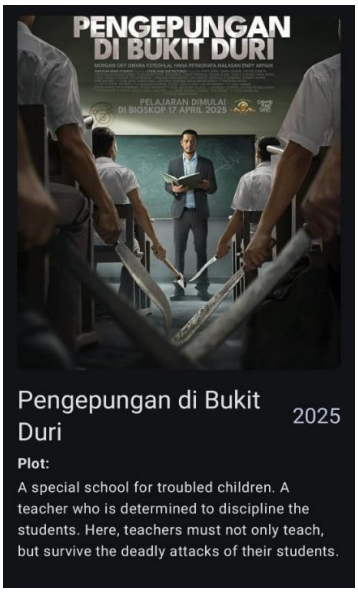
   responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang
7. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
8. Aplikasi berbasis XML harus menggunakan ViewBinding

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



Gambar 8. 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 9. Contoh UI Detail

A. Source Code

1. MainActivity.kt

Tabel 5. Source Code MainActivity.kt

```
1   package com.example.wuwalist
2
3   import android.os.Bundle
4   import android.util.Log
5   import androidx.activity.enableEdgeToEdge
6   import androidx.appcompat.app.AppCompatActivity
7   import androidx.core.view.ViewCompat
8   import androidx.core.view.WindowInsetsCompat
9
10  class MainActivity : AppCompatActivity() {
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14
15          val fragmentManager = supportFragmentManager
16          val homeFragment = HomeFragment()
17          val fragment =
    fragmentManager.findFragmentByTag(HomeFragment::class.ja
    va.simpleName)
18          if (fragment !is HomeFragment) {
19              Log.d("MyFlexibleFragment", "Fragment Name
    :" + HomeFragment::class.java.simpleName)
```

```
            fragmentManager
                .beginTransaction()
                .add(R.id.frame_container, homeFragment,
HomeFragment::class.java.simpleName)
                .commit()
        }
    }
}
```

2. Character.kt

Tabel 6. Source Code Character.kt

```
package com.example.wuwalist

import android.os.Parcelable
import kotlinx.parcelize.Parcelize


@Parcelize
data class Character(
    val name: String,
    val link: String,
    val description: String,
    val photo: Int
):Parcelable
```

3. ListCharacterAdapter.kt

Tabel 7. Source Code ListCharacterAdapter.kt

```
package com.example.wuwalist

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView

class ListCharacterAdapter(
    private val listCharacter: ArrayList<Character>,
    private val onWikiClick: (String) -> Unit,
    private val onDetailClick: (String, Int, String) ->
Unit)
    :
RecyclerView.Adapter<ListCharacterAdapter.ListViewHolder
>() {
```

```
16
17      class ListViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
18          val imgPhoto: ImageView =
    itemView.findViewById(R.id.img_character)
19          val tvName: TextView =
    itemView.findViewById(R.id.tv_character)
20          val tvDeskripsi: TextView =
    itemView.findViewById(R.id.tv_deskripsi)
21          val tvProfile: TextView =
    itemView.findViewById(R.id.tv_deskripsi)
22          val btnWiki: Button =
    itemView.findViewById(R.id.btn_link)
23          val btnDetail: Button =
    itemView.findViewById(R.id.btn_detail)
24      }
25
26      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ListViewHolder {
27          val view: View =
    LayoutInflater.from(parent.context).inflate(R.layout.ite
    m_character, parent, false)
28          return ListViewHolder(view)
29      }
30
31      override fun getItemCount(): Int =
    listCharacter.size
32
33      override fun onBindViewHolder(holder:
    ListViewHolder, position: Int) {
34          val (name, link, description, photo) =
    listCharacter[position]
35          holder.tvName.text = name
36          holder.imgPhoto.setImageResource(photo)
37          holder.tvDeskripsi.text = description
38          holder.btnWiki.setOnClickListener {
    onWikiClick(link) }
39          holder.btnDetail.setOnClickListener {
    onDetailClick(name, photo, description) }
40      }
41  }
```

4. HomeFragment.kt

Tabel 8. Source Code HomeFragment.kt

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

```
42              list,
43              onWikiClick = { link ->
44                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(link))
45                  startActivity(intent)
46              },
47              onDetailClick = { name, photo, description ->
48                  val detailFragment =
    DetailFragment().apply {
49                      arguments = Bundle().apply {
50                          putString("EXTRA_NAME", name)
51                          putInt("EXTRA_PHOTO", photo)
52                          putString("EXTRA_DESCRIPTION",
    description)
53                      }
54                  }
55
56                  parentFragmentManager.beginTransaction()
57                      .replace(R.id.frame_container,
    detailFragment)
58                      .addToBackStack(null)
59                      .commit()
60              }
61          )
62
63          binding.rvCharacter.apply {
64              layoutManager = LinearLayoutManager(context)
65              adapter = characterAdapter
66              setHasFixedSize(true)
67          }
68      }
69
70      private fun getListCharacter(): ArrayList<Character>
    {
71          val dataName =
    resources.getStringArray(R.array.nama_character)
72          val dataPhoto =
    resources.obtainTypedArray(R.array.photo_character)
73          val dataLink =
    resources.getStringArray(R.array.link_character)
74          val dataDescription =
    resources.getStringArray(R.array.deskripsi_character)
75          val listCharacter = ArrayList<Character>()
76          for (i in dataName.indices) {
77              val character = Character(dataName[i],
    dataLink[i], dataDescription[i],
    dataPhoto.getResourceId(i, -1))
78              listCharacter.add(character)
79          }
```

```
        dataPhoto.recycle()
        return listCharacter
    }

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

5. DetailFragment.kt

Tabel 9. Source Code DetailFragment.kt

```
package com.example.wuwalist

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import
com.example.wuwalist.databinding.FragmentDetailBinding

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)

        val name = arguments?.getString("EXTRA_NAME")
        val photo = arguments?.getInt("EXTRA_PHOTO")
        val description =
arguments?.getString("EXTRA_DESCRIPTION")


        binding.tvCharacter.text = name
        binding.tvDeskripsi.text = description
        photo?.let {
            binding.imgCharacter.setImageResource(it)
        }

        return binding.root
```

```
33      }
34
35      override fun onDestroyView() {
36          super.onDestroyView()
37          _binding = null
38      }
39  }
```

6. activity_main.xml

Tabel 10. Source Code activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
    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=".MainActivity"
8       android:id="@+id/frame_container">
9   </FrameLayout>
```

7. fragment_home.xml

Tabel 11. Source Code fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="wrap_content"
5       android:layout_height="wrap_content"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:orientation="horizontal"
8       tools:context=".HomeFragment">
9
10      <androidx.recyclerview.widget.RecyclerView
11          android:id="@+id/rv_character"
12          android:layout_width="match_parent"
13          android:layout_height="match_parent"
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>
```

8. fragment_detail.xml

Tabel 12. Source Code fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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=".DetailFragment">
8
9       <ImageView
10          android:id="@+id/img_character"
11          android:layout_width="190dp"
12          android:layout_height="261dp"
13          android:layout_marginTop="16dp"
14          android:src="@drawable/card_carlotta"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18
19      <TextView
20          android:id="@+id/tv_character"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:layout_marginTop="19dp"
24          android:text="TextView"
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27
    app:layout_constraintTop_toBottomOf="@+id/img_character"
    />
28
29      <TextView
30          android:id="@+id/tv_profile"
31          android:layout_width="wrap_content"
32          android:layout_height="wrap_content"
33          android:layout_marginStart="10dp"
34          android:gravity="start"
35          android:text="Profile"
36          android:textStyle="bold"
37          app:layout_constraintStart_toStartOf="parent"
38
    app:layout_constraintTop_toBottomOf="@+id/tv_character"
    />
39
40      <TextView
```

```
41          android:id="@+id/tv_deskripsi"
42          android:layout_width="wrap_content"
43          android:layout_height="wrap_content"
44          android:layout_marginStart="10dp"
45          android:layout_marginTop="20dp"
46          android:text="TextView"
47          app:layout_constraintStart_toStartOf="parent"
48
    app:layout_constraintTop_toBottomOf="@+id/tv_profile" />
49  </androidx.constraintlayout.widget.ConstraintLayout>
```

9. item_character.xml

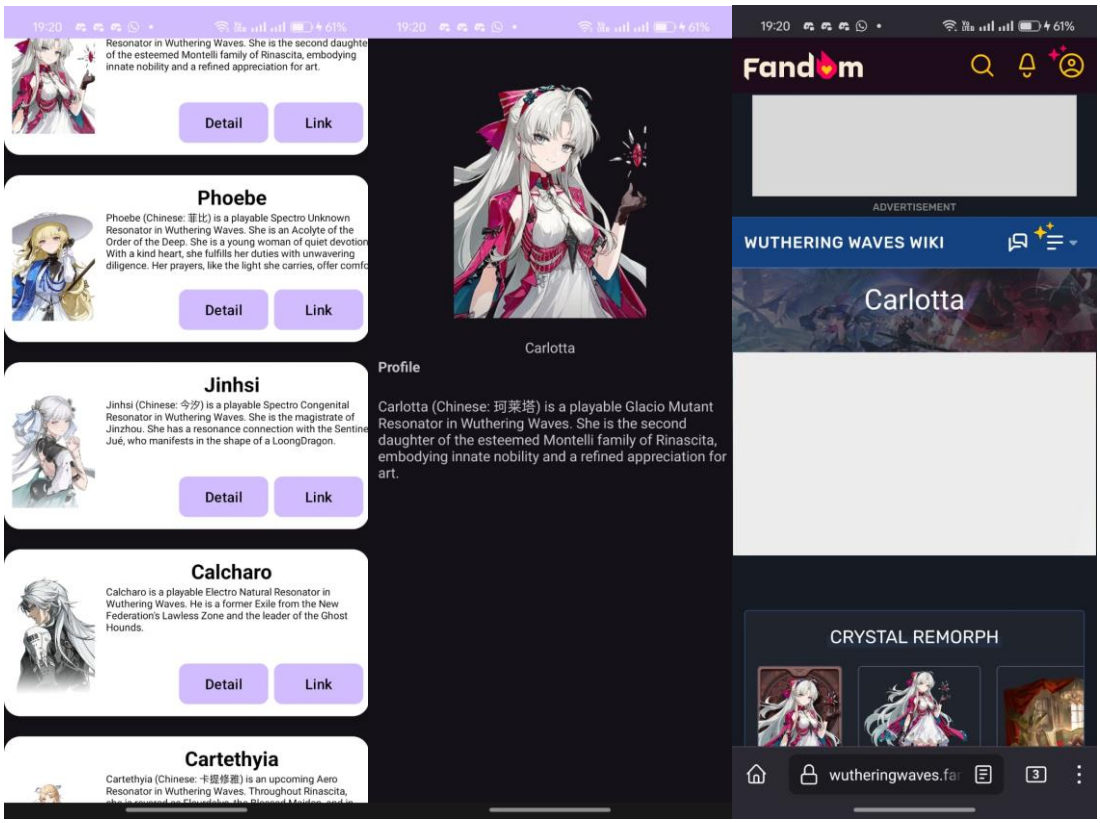
Tabel 13. Source Code item_character.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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="wrap_content">
7
8       <androidx.cardview.widget.CardView
9           android:id="@+id/card_view"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:layout_marginStart="20dp"
13          android:layout_marginTop="10dp"
14          android:layout_marginEnd="20dp"
15          android:layout_marginBottom="10dp"
16          app:cardBackgroundColor="#FFFFFF"
17          app:cardCornerRadius="16dp"
18          app:cardElevation="16dp"
19          app:cardPreventCornerOverlap="false"
20          app:layout_constraintBottom_toBottomOf="parent"
21          app:layout_constraintEnd_toEndOf="parent"
22          app:layout_constraintStart_toStartOf="parent"
23          app:layout_constraintTop_toTopOf="parent">
24
25
    <androidx.constraintlayout.widget.ConstraintLayout
26              android:layout_width="match_parent"
27              android:layout_height="match_parent"
28              android:layout_marginStart="8dp"
29              android:layout_marginTop="8dp"
30              android:layout_marginBottom="8dp">
31
```

```
32              <TextView
33                  android:id="@+id/tv_character"
34                  android:layout_width="wrap_content"
35                  android:layout_height="wrap_content"
36                  android:text="Carlotta"
37                  android:textColor="#000000"
38                  android:textSize="20sp"
39                  android:textStyle="bold"
40
    app:layout_constraintEnd_toEndOf="parent"
41
    app:layout_constraintStart_toEndOf="@+id/img_character"
42
    app:layout_constraintTop_toTopOf="parent" />
43
44              <ImageView
45                  android:id="@+id/img_character"
46                  android:layout_width="90dp"
47                  android:layout_height="143dp"
48
    app:layout_constraintBottom_toBottomOf="parent"
49
    app:layout_constraintEnd_toStartOf="@+id/tv_deskripsi"
50
    app:layout_constraintHorizontal_bias="0.4"
51
    app:layout_constraintStart_toStartOf="parent"
52
    app:layout_constraintTop_toTopOf="parent"
53
    app:layout_constraintVertical_bias="0.526"
54                  app:srcCompat="@drawable/card_carlotta"
    />
55
56              <TextView
57                  android:id="@+id/tv_deskripsi"
58                  android:layout_width="268dp"
59                  android:layout_height="62dp"
60                  android:layout_marginStart="11dp"
61                  android:text="Carlotta (Chinese: 珂莱塔)
    is a playable Glacio Mutant Resonator in Wuthering
    Waves. She is the second daughter of the esteemed
    Montelli family of Rinascita, embodying innate nobility
    and a refined appreciation for art."
62                  android:textColor="#000000"
63                  android:textSize="10sp"
64
    app:layout_constraintEnd_toEndOf="parent"
```

```
65
    app:layout_constraintStart_toEndOf="@+id/img_character"
66
    app:layout_constraintTop_toBottomOf="@+id/tv_character"
    />
67
68              <Button
69                  android:id="@+id/btn_detail"
70                  android:layout_width="wrap_content"
71                  android:layout_height="wrap_content"
72                  android:layout_marginStart="83dp"
73                  android:text="Detail"
74                  android:textColor="#000000"
75                  app:cornerRadius="8dp"
76
    app:layout_constraintBottom_toBottomOf="parent"
77
    app:layout_constraintStart_toEndOf="@+id/img_character"
78
    app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
79                  app:layout_constraintVertical_bias="1.0"
    />
80
81              <Button
82                  android:id="@+id/btn_link"
83                  android:layout_width="wrap_content"
84                  android:layout_height="wrap_content"
85                  android:layout_marginStart="20dp"
86                  android:layout_marginTop="12dp"
87                  android:layout_marginEnd="20dp"
88                  android:text="Link"
89                  android:textColor="#000000"
90                  app:cornerRadius="8dp"
91
    app:layout_constraintBottom_toBottomOf="parent"
92
    app:layout_constraintEnd_toEndOf="parent"
93
    app:layout_constraintStart_toEndOf="@+id/btn_detail"
94
    app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
95                  app:layout_constraintVertical_bias="0.0"
    />
96
    </androidx.constraintlayout.widget.ConstraintLayout>
97      </androidx.cardview.widget.CardView>
98  </androidx.constraintlayout.widget.ConstraintLayout>
```

B. Output Program



Gambar 10. Screenshot Hasil Jawaban Soal 1 Modul 3

C. Pembahasan

1. MainActivity.kt:

- **Pada baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan, termasuk komponen dasar Android seperti Bundle, AppCompatActivity, dan utilitas untuk mengelola fragment. Impor ini memungkinkan aplikasi untuk menggunakan fitur-fitur Android modern seperti Fragment Manager dan logging.

- **Pada baris 11**, didefinisikan kelas MainActivity yang mewarisi AppCompatActivity, yang merupakan kelas dasar untuk aktivitas yang mendukung fitur-fitur kompatibilitas Android.
- **Pada baris 12 hingga 14**, terdapat method onCreate() yang merupakan entry point utama aktivitas. Di sini, layout activity_main.xml diinisialisasi menggunakan setContentView() untuk menampilkan antarmuka pengguna.
- **Pada baris 16 hingga 17**, dilakukan inisialisasi FragmentManager dan pembuatan instance HomeFragment. FragmentManager digunakan untuk mengelola fragment-fragment dalam aplikasi.
- **Pada baris 18**, dilakukan pengecekan apakah HomeFragment sudah ada dalam container menggunakan findFragmentByTag(). Ini mencegah duplikasi fragment saat aktivitas di-recreate, misalnya saat rotasi layar.
- **Pada baris 19 hingga 24**, terdapat blok kondisional yang hanya dijalankan jika HomeFragment belum ada. Di dalamnya, log debugging ditambahkan dan transaction fragment dilakukan untuk menambahkan HomeFragment ke container dengan ID frame_container.

2. Character.kt:

- **Pada baris 1 hingga 4**, dilakukan impor library yang diperlukan untuk membuat kelas data yang dapat di-parcel, yaitu Parcelable dan anotasi Parcelize.
- **Pada baris 6 hingga 12**, didefinisikan data class Character dengan anotasi @Parcelize yang mengimplementasikan interface Parcelable. Kelas ini memiliki empat properti: name (nama karakter), link (URL wiki), description (deskripsi karakter), dan photo (ID resource gambar).
- Penggunaan anotasi @Parcelize **pada baris 6** memungkinkan Kotlin untuk secara otomatis menghasilkan implementasi Parcelable tanpa perlu menulis kode boilerplate, sehingga memudahkan pengiriman objek Character antar komponen Android.

3. **ListCharacterAdapter.kt:**

- **Pada baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan untuk membuat adapter RecyclerView, termasuk komponen View, ViewGroup, dan widget-widget Android seperti Button, ImageView, dan TextView.
- **Pada baris 11 hingga 15**, didefinisikan kelas ListCharacterAdapter yang mewarisi RecyclerView.Adapter dengan tipe parameter ListViewHolder. Adapter ini menerima tiga parameter: listCharacter (daftar karakter), onWikiClick (callback untuk klik tombol wiki), dan onDetailClick (callback untuk klik tombol detail).
- **Pada baris 17 hingga 24**, didefinisikan inner class ListViewHolder yang mewarisi RecyclerView.ViewHolder. Kelas ini menginisialisasi referensi ke semua view dalam layout item_character menggunakan findViewById().
- **Pada baris 26 hingga 29**, diimplementasikan method onCreateViewHolder() yang meng-inflate layout item_character.xml dan mengembalikan instance ListViewHolder baru.
- **Pada baris 31**, diimplementasikan method getItemCount() yang mengembalikan jumlah item dalam listCharacter.
- **Pada baris 33 hingga 40**, diimplementasikan method onBindViewHolder() yang mengisi data ke view-view dalam ViewHolder. Pada baris 34, digunakan destructuring declaration untuk mengekstrak properti dari objek Character. **Pada baris 35-37**, data diset ke view yang sesuai. **Pada baris 38-39**, listener dikonfigurasi untuk tombol Wiki dan Detail.

4. **HomeFragment.kt:**

- **Pada baris 1 hingga 14**, dilakukan impor berbagai library yang diperlukan, termasuk Intent, Uri, Fragment, RecyclerView, dan komponen View Binding.

- **Pada baris 16 hingga 23**, didefinisikan kelas HomeFragment yang mewarisi Fragment. Di dalamnya, terdapat deklarasi variabel binding dengan pattern yang aman untuk memory leak, serta variabel adapter dan list untuk menyimpan data karakter.
- **Pada baris 25 hingga 36**, diimplementasikan method onCreateView() yang meng-inflate layout fragment_home.xml menggunakan View Binding. **Pada baris 31-32**, list karakter diisi dan RecyclerView diatur.
- **Pada baris 38 hingga 65**, didefinisikan method setupRecyclerView() yang menginisialisasi adapter dengan dua callback:
    - **Pada baris 41-44**, callback onWikiClick membuka browser dengan URL karakter.
    - **Pada baris 45-59**, callback onDetailClick membuat instance DetailFragment, mengisi arguments-nya dengan data karakter, dan melakukan transaction fragment untuk menampilkan detail.
    - **Pada baris 61-64**, RecyclerView dikonfigurasi dengan LinearLayoutManager dan adapter yang telah dibuat.
- **Pada baris 67 hingga 80**, didefinisikan method getListCharacter() yang mengambil data karakter dari resources. **Pada baris 68-71**, array data diambil dari resources. **Pada baris 72-76**, objek Character dibuat untuk setiap set data dan ditambahkan ke list. **Pada baris 77**, TypedArray di-recycle untuk membebaskan resources.
- **Pada baris 82 hingga 85**, diimplementasikan method onDestroyView() yang membersihkan referensi binding untuk mencegah memory leak.

5. DetailFragment.kt:

- **Pada baris 1 hingga 7**, dilakukan impor library yang diperlukan untuk membuat fragment dan menggunakan View Binding.
- **Pada baris 9 hingga 13**, didefinisikan kelas DetailFragment yang mewarisi Fragment. Di dalamnya, terdapat deklarasi variabel binding dengan pattern yang aman untuk memory leak.

- **Pada baris 15 hingga 32**, diimplementasikan method onCreateView() yang meng-inflate layout fragment_detail.xml menggunakan View Binding. **Pada baris 20-22**, data karakter diambil dari arguments. **Pada baris 24-28**, data tersebut ditampilkan pada view-view yang sesuai.
- **Pada baris 34 hingga 37**, diimplementasikan method onDestroyView() yang membersihkan referensi binding untuk mencegah memory leak.

6. **activity_main.xml:**

- **Pada baris 1 hingga 9**, didefinisikan layout utama aplikasi menggunakan FrameLayout. Layout ini diberi ID frame_container **pada baris 8**, yang digunakan sebagai container untuk fragment-fragment dalam aplikasi.

7. **fragment_home.xml:**

- **Pada baris 1 hingga 8**, didefinisikan layout untuk HomeFragment menggunakan ConstraintLayout. Layout ini dikonfigurasi dengan lebar dan tinggi wrap_content, yang mungkin perlu diubah menjadi match_parent untuk mengisi seluruh layar.
- **Pada baris 10 hingga 16**, didefinisikan RecyclerView dengan ID rv_character yang akan menampilkan daftar karakter. RecyclerView ini dikonfigurasi untuk mengisi seluruh parent layout menggunakan constraint.

8. **fragment_detail.xml:**

- **Pada baris 1 hingga 7**, didefinisikan layout untuk DetailFragment menggunakan ConstraintLayout dengan lebar dan tinggi match_parent.
- **Pada baris 9 hingga 16**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di tengah atas layout.
- **Pada baris 18 hingga 25**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini diposisikan di bawah gambar dan di tengah horizontal.

- **Pada baris 27 hingga 35**, didefinisikan TextView dengan ID tv_profile sebagai label untuk bagian profil. TextView ini diposisikan di bawah nama dengan alignment kiri dan style bold.
- **Pada baris 37 hingga 44**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini diposisikan di bawah label profil dengan alignment kiri.

9. item_character.xml:

- **Pada baris 1 hingga 6**, didefinisikan layout root untuk item karakter menggunakan ConstraintLayout dengan lebar match_parent dan tinggi wrap_content.
- **Pada baris 8 hingga 24**, didefinisikan CardView dengan ID card_view yang memberikan efek card dengan sudut melengkung dan bayangan. CardView ini dikonfigurasi dengan berbagai properti seperti margin, warna latar, radius sudut, dan elevasi.
- **Pada baris 26 hingga 30**, didefinisikan ConstraintLayout di dalam CardView yang akan menampung semua elemen UI item karakter.
- **Pada baris 32 hingga 42**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini dikonfigurasi dengan style bold, ukuran 20sp, dan warna teks hitam.
- **Pada baris 44 hingga 55**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di sebelah kiri layout.
- **Pada baris 57 hingga 69**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini dikonfigurasi dengan ukuran 10sp dan warna teks hitam.
- **Pada baris 71 hingga 82**, didefinisikan Button dengan ID btn_detail untuk melihat detail karakter. Button ini dikonfigurasi dengan sudut melengkung dan warna teks hitam.

- **Pada baris 84 hingga 96**, didefinisikan Button dengan ID btn_link untuk membuka link karakter di browser. Button ini juga dikonfigurasi dengan sudut melengkung dan warna teks hitam.

SOAL 2

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

A. Jawaban

RecyclerView memang sering dianggap memiliki kode yang panjang dan cenderung repetitif atau boilerplate, apalagi jika dibandingkan dengan komponen seperti LazyColumn di Jetpack Compose yang jauh lebih ringkas dan modern. Namun, dalam konteks aplikasi berbasis XML seperti proyek saya, penggunaan RecyclerView masih menjadi pilihan yang relevan dan masuk akal.

Pertama, RecyclerView sudah lama menjadi standar dalam pengembangan UI berbasis daftar di Android. Komponen ini sudah sangat stabil, banyak didukung oleh pustaka eksternal, dan terdokumentasi dengan sangat baik. Hal ini membuatnya tetap menjadi pilihan utama, terutama bagi pengembang yang masih menggunakan pendekatan View System tradisional. Dalam proyek saya, misalnya, seluruh tampilan masih berbasis XML, sehingga lebih masuk akal untuk menggunakan RecyclerView dibanding harus mengadopsi Jetpack Compose hanya untuk mengganti daftar tampilan.

Selain itu, RecyclerView juga memberi kontrol yang lebih lengkap terhadap tampilan item dan bagaimana daftar tersebut ditampilkan atau diatur. Fitur-fitur seperti animasi, pengaturan cache, serta berbagai jenis layout manager menjadi nilai lebih yang membuatnya fleksibel untuk berbagai kebutuhan UI. Meskipun lebih verbose, struktur RecyclerView memberikan kejelasan dalam pemisahan logika tampilan dan data, khususnya dalam skala proyek yang lebih besar.

Jadi, meskipun secara sintaksis LazyColumn terlihat lebih sederhana dan modern, RecyclerView masih memiliki tempat penting, terutama ketika kita bekerja dalam lingkungan XML atau saat memerlukan fleksibilitas dan kontrol penuh terhadap tampilan daftar.

MODUL 4 :
ViewModel and Debugging
SOAL 1

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

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

A. Source Code

1. MainActivity.kt

Tabel 14. Source Code MainActivity.kt

```
1  package com.example.wuwalist
2
3  import android.os.Bundle
4  import android.util.Log
5  import androidx.activity.enableEdgeToEdge
6  import androidx.appcompat.app.AppCompatActivity
```

```
7   import androidx.core.view.ViewCompat
8   import androidx.core.view.WindowInsetsCompat
9
10  class MainActivity : AppCompatActivity() {
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          setContentView(R.layout.activity_main)
14
15          val fragmentManager = supportFragmentManager
16          val homeFragment = HomeFragment()
17          val fragment =
    fragmentManager.findFragmentByTag(HomeFragment::class.ja
    va.simpleName)
18          if (fragment !is HomeFragment) {
19              Log.d("MyFlexibleFragment", "Fragment Name
    :" + HomeFragment::class.java.simpleName)
20              fragmentManager
21                  .beginTransaction()
22                  .add(R.id.frame_container, homeFragment,
    HomeFragment::class.java.simpleName)
23                  .commit()
24          }
25      }
26  }
```

2. Character.kt

Tabel 15. Source Code Character.kt

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

3. ListCharacterAdapter.kt

Tabel 16. Source Code ListCharacterAdapter.kt

```
1   package com.example.wuwalist.adapter
2
3   import android.view.LayoutInflater
4   import android.view.View
5   import android.view.ViewGroup
6   import android.widget.Button
7   import android.widget.ImageView
8   import android.widget.TextView
9   import androidx.recyclerview.widget.RecyclerView
10  import com.example.wuwalist.model.Character
11  import com.example.wuwalist.R
12
13  class ListCharacterAdapter(
14      private val listCharacter: ArrayList<Character>,
15      private val onWikiClick: (String) -> Unit,
16      private val onDetailClick: (String, Int, String) ->
    Unit)
17      :
    RecyclerView.Adapter<ListCharacterAdapter.ListViewHolder
    >() {
18
19      class ListViewHolder(itemView: View) :
    RecyclerView.ViewHolder(itemView) {
20          val imgPhoto: ImageView =
    itemView.findViewById(R.id.img_character)
21          val tvName: TextView =
    itemView.findViewById(R.id.tv_character)
22          val tvDeskripsi: TextView =
    itemView.findViewById(R.id.tv_deskripsi)
23          val btnWiki: Button =
    itemView.findViewById(R.id.btn_link)
24          val btnDetail: Button =
    itemView.findViewById(R.id.btn_detail)
25      }
26
27      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ListViewHolder {
28          val view: View =
    LayoutInflater.from(parent.context).inflate(R.layout.ite
    m_character, parent, false)
29          return ListViewHolder(view)
30      }
31
```

```
32      override fun getItemCount(): Int =
    listCharacter.size
33
34      override fun onBindViewHolder(holder:
    ListViewHolder, position: Int) {
35          val (name, link, description, photo) =
    listCharacter[position]
36          holder.tvName.text = name
37          holder.imgPhoto.setImageResource(photo)
38          holder.tvDeskripsi.text = description
39          holder.btnWiki.setOnClickListener {
    onWikiClick(link) }
40          holder.btnDetail.setOnClickListener {
    onDetailClick(name, photo, description) }
41      }
42      fun setData(newList: List<Character>) {
43          listCharacter.clear()
44          listCharacter.addAll(newList)
45      }
46  }
```

4. HomeFragment.kt

Tabel 17. Source Code HomeFragment.kt

```
1   package com.example.wuwalist.presentation.home
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import android.util.Log
7   import androidx.fragment.app.Fragment
8   import androidx.recyclerview.widget.LinearLayoutManager
9   import android.view.LayoutInflater
10  import android.view.View
11  import android.view.ViewGroup
12  import androidx.fragment.app.viewModels
13  import androidx.lifecycle.lifecycleScope
14  import com.example.wuwalist.R
15  import
    com.example.wuwalist.adapter.ListCharacterAdapter
16  import
    com.example.wuwalist.databinding.FragmentHomeBinding
17  import com.example.wuwalist.utils.HomeViewModelFactory
18  import
    com.example.wuwalist.presentation.detail.DetailFragment
19  import kotlinx.coroutines.launch
20  import kotlinx.coroutines.flow.collectLatest
21  import java.util.ArrayList
```

```
22
23  class HomeFragment : Fragment() {
24
25      private var _binding: FragmentHomeBinding? = null
26      private val binding get() = _binding!!
27
28      private lateinit var characterAdapter:
    ListCharacterAdapter
29
30      private val viewModel: HomeViewModel by viewModels
    {
31          HomeViewModelFactory(resources)
32      }
33
34      override fun onCreateView(
35          inflater: LayoutInflater, container:
    ViewGroup?,
36          savedInstanceState: Bundle?
37      ): View {
38          _binding =
    FragmentHomeBinding.inflate(inflater, container, false)
39          return binding.root
40      }
41
42      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
43          super.onViewCreated(view, savedInstanceState)
44
45          setupRecyclerView()
46          observeCharacterList()
47
48          viewModel.loadCharacters()
49      }
50
51      private fun setupRecyclerView() {
52          characterAdapter = ListCharacterAdapter(
53              ArrayList(),
54              onWikiClick = { link ->
55
56                  Log.e("Explicit intent to Web","Going
    to $link")
57
58                  val intent = Intent(Intent.ACTION_VIEW,
    Uri.parse(link))
59                  startActivity(intent)
60
61              },
62              onDetailClick = { name, photo, description
    ->
```

```
63                  val detailFragment =
    DetailFragment().apply {
64
65
66
67                      Log.e("Move to detail page","move
    to $name detail page")
68
69                      arguments = Bundle().apply {
70                          putString("EXTRA_NAME", name)
71                          putInt("EXTRA_PHOTO", photo)
72                          putString("EXTRA_DESCRIPTION",
    description)
73                      }
74                  }
75
76
    parentFragmentManager.beginTransaction()
77                      .replace(R.id.frame_container,
    detailFragment)
78                      .addToBackStack(null)
79                      .commit()
80              }
81          )
82
83          binding.rvCharacter.apply {
84              layoutManager =
    LinearLayoutManager(context)
85              adapter = characterAdapter
86              setHasFixedSize(true)
87          }
88      }
89
90      private fun observeCharacterList() {
91          viewLifecycleOwner.lifecycleScope.launch {
92              viewModel.characterList.collectLatest {
    list ->
93                  characterAdapter.setData(list)
94              }
95          }
96      }
97
98      override fun onDestroyView() {
99          super.onDestroyView()
100         _binding = null
101     }
102 }
```

5. DetailFragment.kt

Tabel 18. Source Code DetailFragment.kt

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

6. activity_main.xml

Tabel 19. Source Code activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <FrameLayout
   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=".MainActivity"
8      android:id="@+id/frame_container">
9  </FrameLayout>
```

7. fragment_home.xml

Tabel 20. Source Code fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="wrap_content"
5       android:layout_height="wrap_content"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:orientation="horizontal"
8       tools:context=".HomeFragment">
9
10      <androidx.recyclerview.widget.RecyclerView
11          android:id="@+id/rv_character"
12          android:layout_width="match_parent"
13          android:layout_height="match_parent"
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>
```

8. fragment_detail.xml

Tabel 21. Source Code fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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=".DetailFragment">
8
9       <ImageView
10          android:id="@+id/img_character"
11          android:layout_width="190dp"
12          android:layout_height="261dp"
13          android:layout_marginTop="16dp"
14          android:src="@drawable/card_carlotta"
15          app:layout_constraintEnd_toEndOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintTop_toTopOf="parent" />
18
19      <TextView
20          android:id="@+id/tv_character"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:layout_marginTop="19dp"
24          android:text="TextView"
25          app:layout_constraintEnd_toEndOf="parent"
26          app:layout_constraintStart_toStartOf="parent"
27
    app:layout_constraintTop_toBottomOf="@+id/img_character"
    />
28
29      <TextView
30          android:id="@+id/tv_profile"
31          android:layout_width="wrap_content"
32          android:layout_height="wrap_content"
33          android:layout_marginStart="10dp"
34          android:gravity="start"
35          android:text="Profile"
36          android:textStyle="bold"
37          app:layout_constraintStart_toStartOf="parent"
38
    app:layout_constraintTop_toBottomOf="@+id/tv_character"
    />
```

```
39
40      <TextView
41          android:id="@+id/tv_deskripsi"
42          android:layout_width="wrap_content"
43          android:layout_height="wrap_content"
44          android:layout_marginStart="10dp"
45          android:layout_marginTop="20dp"
46          android:text="TextView"
47          app:layout_constraintStart_toStartOf="parent"
48
    app:layout_constraintTop_toBottomOf="@+id/tv_profile" />
49  </androidx.constraintlayout.widget.ConstraintLayout>
```

9. item_character.xml

Tabel 22. Source Code item_character.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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="wrap_content">
7
8       <androidx.cardview.widget.CardView
9           android:id="@+id/card_view"
10          android:layout_width="wrap_content"
11          android:layout_height="wrap_content"
12          android:layout_marginStart="20dp"
13          android:layout_marginTop="10dp"
14          android:layout_marginEnd="20dp"
15          android:layout_marginBottom="10dp"
16          app:cardBackgroundColor="#FFFFFF"
17          app:cardCornerRadius="16dp"
18          app:cardElevation="16dp"
19          app:cardPreventCornerOverlap="false"
20          app:layout_constraintBottom_toBottomOf="parent"
21          app:layout_constraintEnd_toEndOf="parent"
22          app:layout_constraintStart_toStartOf="parent"
23          app:layout_constraintTop_toTopOf="parent">
24
25
    <androidx.constraintlayout.widget.ConstraintLayout
26              android:layout_width="match_parent"
27              android:layout_height="match_parent"
28              android:layout_marginStart="8dp"
29              android:layout_marginTop="8dp"
```

```
30              android:layout_marginBottom="8dp">
31
32              <TextView
33                  android:id="@+id/tv_character"
34                  android:layout_width="wrap_content"
35                  android:layout_height="wrap_content"
36                  android:text="Carlotta"
37                  android:textColor="#000000"
38                  android:textSize="20sp"
39                  android:textStyle="bold"
40
    app:layout_constraintEnd_toEndOf="parent"
41
    app:layout_constraintStart_toEndOf="@+id/img_character"
42
    app:layout_constraintTop_toTopOf="parent" />
43
44              <ImageView
45                  android:id="@+id/img_character"
46                  android:layout_width="90dp"
47                  android:layout_height="143dp"

48  app:layout_constraintBottom_toBottomOf="parent"

49  app:layout_constraintEnd_toStartOf="@+id/tv_deskripsi"

50  app:layout_constraintHorizontal_bias="0.4"

51  app:layout_constraintStart_toStartOf="parent"

52  app:layout_constraintTop_toTopOf="parent"

53  app:layout_constraintVertical_bias="0.526"
                  app:srcCompat="@drawable/card_carlotta"
54  />

55              <TextView
56                  android:id="@+id/tv_deskripsi"
57                  android:layout_width="268dp"
58                  android:layout_height="62dp"
59                  android:layout_marginStart="11dp"
60                  android:text="Carlotta (Chinese: 珂莱塔)
61  is a playable Glacio Mutant Resonator in Wuthering
    Waves. She is the second daughter of the esteemed
    Montelli family of Rinascita, embodying innate nobility
    and a refined appreciation for art."
                  android:textColor="#000000"
62                android:textSize="10sp"
63
```

```
64
    app:layout_constraintEnd_toEndOf="parent"
65
    app:layout_constraintStart_toEndOf="@+id/img_character"
66
    app:layout_constraintTop_toBottomOf="@+id/tv_character"
    />
67
68              <Button
69                  android:id="@+id/btn_detail"
70                  android:layout_width="wrap_content"
71                  android:layout_height="wrap_content"
72                  android:layout_marginStart="83dp"
73                  android:text="Detail"
74                  android:textColor="#000000"
75                  app:cornerRadius="8dp"
76
    app:layout_constraintBottom_toBottomOf="parent"
77
    app:layout_constraintStart_toEndOf="@+id/img_character"
78
    app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
79                  app:layout_constraintVertical_bias="1.0"
    />
80
81              <Button
82                  android:id="@+id/btn_link"
83                  android:layout_width="wrap_content"
84                  android:layout_height="wrap_content"
85                  android:layout_marginStart="20dp"
86                  android:layout_marginTop="12dp"
87                  android:layout_marginEnd="20dp"
88                  android:text="Link"
89                  android:textColor="#000000"
90                  app:cornerRadius="8dp"
91
    app:layout_constraintBottom_toBottomOf="parent"
92
    app:layout_constraintEnd_toEndOf="parent"
93
    app:layout_constraintStart_toEndOf="@+id/btn_detail"
94
    app:layout_constraintTop_toBottomOf="@+id/tv_deskripsi"
95                  app:layout_constraintVertical_bias="0.0"
    />
96
    </androidx.constraintlayout.widget.ConstraintLayout>
97      </androidx.cardview.widget.CardView>
98  </androidx.constraintlayout.widget.ConstraintLayout>
```

10. HomeViewModel.kt

Tabel 23. Source Code HomeViewModel.kt

```
1   package com.example.wuwalist.presentation.home
2
3   import android.content.res.Resources
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import com.example.wuwalist.R
7   import com.example.wuwalist.model.Character
8   import kotlinx.coroutines.flow.Flow
9   import kotlinx.coroutines.flow.MutableStateFlow
10  import kotlinx.coroutines.flow.StateFlow
11  import kotlinx.coroutines.flow.flow
12  import kotlinx.coroutines.flow.onStart
13  import kotlinx.coroutines.launch
14
15  class HomeViewModel(private val resources: Resources) :
    ViewModel() {
16
17      private         val         _characterList         =
    MutableStateFlow<List<Character>>(emptyList())
18      val characterList: StateFlow<List<Character>> get() =
    _characterList
19
20      private fun getCharacterFlow(): Flow<List<Character>>
    = flow {
21
22          val                  dataName                  =
    resources.getStringArray(R.array.nama_character)
23          val                  dataLink                  =
    resources.getStringArray(R.array.link_character)
24          val               dataDescription              =
    resources.getStringArray(R.array.deskripsi_character)
25          val                 dataPhoto                  =
    resources.obtainTypedArray(R.array.photo_character)
26
27          val listCharacter = ArrayList<Character>()
28          for (i in dataName.indices) {
29              val    chara    =    Character(dataName[i],
    dataLink[i],
    dataDescription[i],dataPhoto.getResourceId(i, -1))
30              listCharacter.add(chara)
31          }
32          dataPhoto.recycle()
33          emit(listCharacter)
34      }
```

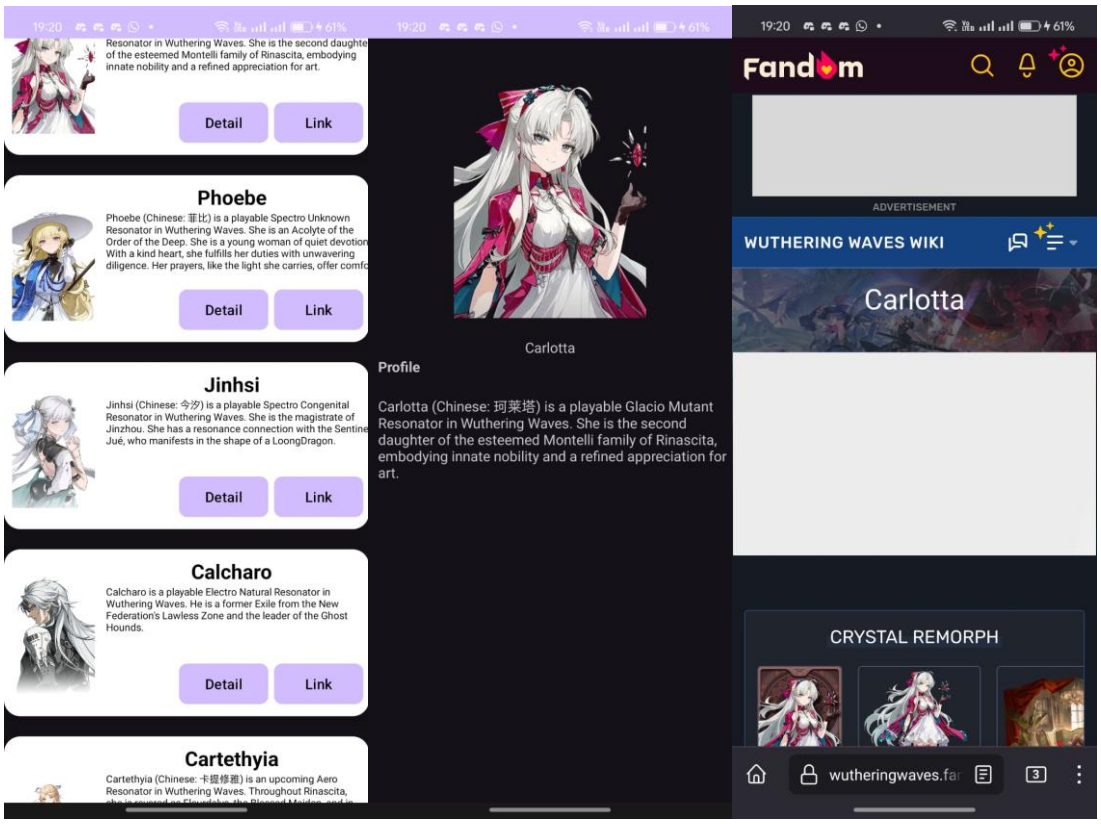
```
35
36      fun loadCharacters() {
37          viewModelScope.launch {
38              getCharacterFlow()
39                  .onStart {
40                      _characterList.value = emptyList()
41                  }
42                  .collect { characters ->
43                      _characterList.value = characters
44                  }
45          }
46      }
47  }
```

11. ViewModelFactory.kt

Tabel 24. Source Code ViewModelFactory.kt

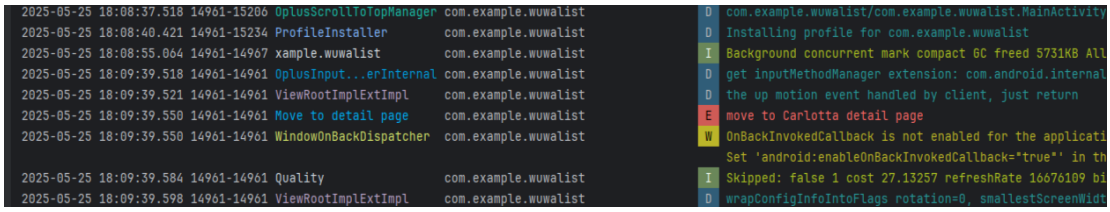
```
1   package com.example.wuwalist.utils
2
3   import android.content.res.Resources
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6   import
    com.example.wuwalist.presentation.home.HomeViewModel
7
8   class    HomeViewModelFactory(private    val    resources:
    Resources) : ViewModelProvider.Factory {
9       override  fun  <T  :  ViewModel>  create(modelClass:
    Class<T>): T {
10          if
    (modelClass.isAssignableFrom(HomeViewModel::class.java))
    {
11              @Suppress("UNCHECKED_CAST")
12              return HomeViewModel(resources) as T
13          }
14          throw IllegalArgumentException("Unknown ViewModel
    class")
15      }
16  }
```

B. Output Program

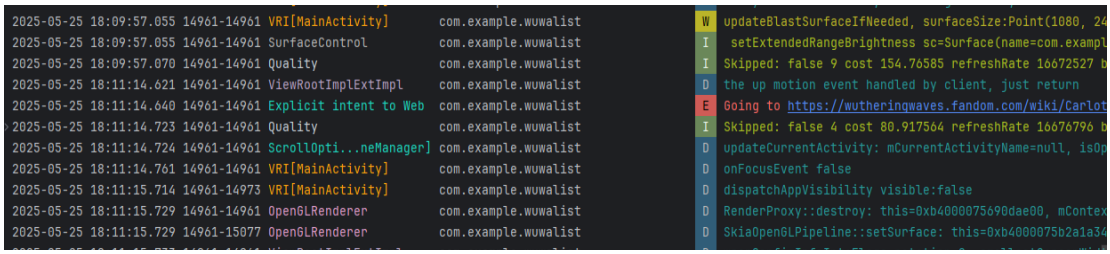


Gambar 11. Screenshot Hasil Jawaban Soal 1 Modul 4

Gambar 12. Screenshot Log Saat Data Item Masuk Ke Dalam List Modul 4



Gambar 13. Screenshot Log Saat Tombol Detail Dan Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Modul 4



Gambar 14. Screenshot Log Tombol Explicit Intent Ditekan Modul 4

C. Pembahasan

1. MainActivity.kt:

- **Pada baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan, termasuk komponen dasar Android seperti Bundle, AppCompatActivity, dan utilitas untuk mengelola fragment. Impor ini memungkinkan aplikasi untuk menggunakan fitur-fitur Android modern seperti Fragment Manager dan logging.
- **Pada baris 11**, didefinisikan kelas MainActivity yang mewarisi AppCompatActivity, yang merupakan kelas dasar untuk aktivitas yang mendukung fitur-fitur kompatibilitas Android.
- **Pada baris 12 hingga 14**, terdapat method onCreate() yang merupakan entry point utama aktivitas. Di sini, layout activity_main.xml diinisialisasi menggunakan setContentView() untuk menampilkan antarmuka pengguna.
- **Pada baris 16 hingga 17**, dilakukan inisialisasi FragmentManager dan pembuatan instance HomeFragment. FragmentManager digunakan untuk mengelola fragment-fragment dalam aplikasi.
- **Pada baris 18**, dilakukan pengecekan apakah HomeFragment sudah ada dalam container menggunakan findFragmentByTag(). Ini mencegah duplikasi fragment saat aktivitas di-recreate, misalnya saat rotasi layar.

- **Pada baris 19 hingga 24**, terdapat blok kondisional yang hanya dijalankan jika HomeFragment belum ada. Di dalamnya, log debugging ditambahkan dan transaction fragment dilakukan untuk menambahkan HomeFragment ke container dengan ID frame_container.

2. **Character.kt:**

- **Pada baris 1 hingga 4**, dilakukan impor library yang diperlukan untuk membuat kelas data yang dapat di-parcel, yaitu Parcelable dan anotasi Parcelize.
- **Pada baris 6 hingga 12**, didefinisikan data class Character dengan anotasi @Parcelize yang mengimplementasikan interface Parcelable. Kelas ini memiliki empat properti: name (nama karakter), link (URL wiki), description (deskripsi karakter), dan photo (ID resource gambar).
- Penggunaan anotasi @Parcelize **pada baris 6** memungkinkan Kotlin untuk secara otomatis menghasilkan implementasi Parcelable tanpa perlu menulis kode boilerplate, sehingga memudahkan pengiriman objek Character antar komponen Android.

3. **ListCharacterAdapter.kt:**

- Pada **baris 1 hingga 9**, dilakukan impor berbagai library yang diperlukan untuk membuat adapter RecyclerView, termasuk komponen View, ViewGroup, dan widget-widget Android seperti Button, ImageView, dan TextView.

- Pada **baris 11 hingga 15**, didefinisikan kelas ListCharacterAdapter yang mewarisi RecyclerView.Adapter dengan tipe parameter ListViewHolder. Adapter ini menerima tiga parameter: listCharacter (sebuah ArrayList dari objek Character), onWikiClick (sebuah fungsi lambda yang akan dipanggil ketika tombol wiki diklik, membawa String sebagai argumen), dan onDetailClick (sebuah fungsi lambda yang akan dipanggil ketika tombol detail diklik, membawa String, Int, dan String sebagai argumen).
- Pada **baris 17 hingga 24**, didefinisikan inner class ListViewHolder yang mewarisi RecyclerView.ViewHolder. Kelas ini bertugas untuk menyimpan referensi ke view yang ada di dalam setiap item RecyclerView. Inisialisasi view (imgPhoto, tvName, tvDeskripsi, btnWiki, btnDetail) dilakukan menggunakan itemView.findViewById().
- Pada **baris 26 hingga 29**, diimplementasikan method onCreateViewHolder(). Method ini dipanggil ketika RecyclerView membutuhkan ViewHolder baru. Di sini, layout R.layout.item_character di-inflate (diubah dari XML menjadi objek View) menggunakan LayoutInflater dan kemudian sebuah instance ListViewHolder baru dibuat dan dikembalikan.
- Pada **baris 31**, diimplementasikan method getItemCount() yang mengembalikan jumlah total item dalam listCharacter.
- Pada **baris 33 hingga 40**, diimplementasikan method onBindViewHolder(). Method ini dipanggil oleh RecyclerView untuk menampilkan data pada posisi tertentu.
    - Pada **baris 34**, dilakukan destructuring declaration untuk mengambil properti name, link, description, dan photo dari objek Character pada posisi saat ini di listCharacter.
    - Pada **baris 35 hingga 37**, data karakter (nama, gambar, dan deskripsi) diatur ke TextView dan ImageView yang sesuai di dalam ViewHolder.
    - Pada **baris 38**, sebuah OnClickListener diatur untuk btnWiki. Ketika tombol ini diklik, fungsi lambda onWikiClick akan dipanggil dengan meneruskan link karakter.

- Pada **baris 39**, sebuah OnClickListener diatur untuk btnDetail. Ketika tombol ini diklik, fungsi lambda onDetailClick akan dipanggil dengan meneruskan name, photo, dan description karakter.
- Pada **baris 41 hingga 44**, didefinisikan fungsi setData() yang digunakan untuk memperbarui data di dalam adapter. Fungsi ini akan menghapus data lama dari listCharacter dan menambahkan semua data baru dari newList.

4. HomeFragment.kt:

- Pada **baris 1 hingga 16**, dilakukan impor library yang dibutuhkan, termasuk Fragment, Intent, Uri untuk intent eksplisit, komponen RecyclerView, ViewModel, LifecycleScope untuk coroutines, serta kelas-kelas dari paket proyek seperti ListCharacterAdapter, FragmentHomeBinding, HomeViewModelFactory, dan DetailFragment.
- Pada **baris 18 dan 19**, dideklarasikan variabel _binding dan binding untuk view binding dengan FragmentHomeBinding. Penggunaan !! (not-null asserted call) pada binding mengasumsikan _binding akan selalu terinisialisasi setelah onCreateView dan sebelum onDestroyView.
- Pada **baris 22**, dideklarasikan variabel characterAdapter untuk ListCharacterAdapter.
- Pada **baris 24 hingga 26**, viewModel diinisialisasi menggunakan delegasi viewModels dengan HomeViewModelFactory yang menerima resources sebagai dependensi. Ini memastikan HomeViewModel dibuat dengan benar.
- Pada **baris 28 hingga 33**, method onCreateView() meng-inflate layout FragmentHomeBinding dan mengembalikan binding.root sebagai view untuk fragment.
- Pada **baris 35 hingga 40**, method onViewCreated() dipanggil setelah view untuk fragment dibuat. Di sini, setupRecyclerView() dipanggil untuk menginisialisasi RecyclerView, observeCharacterList() dipanggil untuk mulai mengamati data dari ViewModel, dan viewModel.loadCharacters() dipanggil untuk memuat data karakter.

- Pada **baris 42 hingga 71**, didefinisikan fungsi setupRecyclerView():
    - Pada **baris 43 hingga 65**, characterAdapter diinisialisasi.
        - Parameter ArrayList() dikirim sebagai daftar awal (kosong) untuk adapter.
        - Parameter onWikiClick diisi dengan lambda yang akan membuat Intent dengan ACTION_VIEW untuk membuka link di browser. Terdapat Log.e untuk mencatat event saat tombol ini ditekan.
        - Parameter onDetailClick diisi dengan lambda yang akan membuat instance DetailFragment, menaruh data karakter (name, photo, description) ke dalam Bundle sebagai arguments, dan kemudian melakukan transaksi fragment untuk mengganti konten R.id.frame_container dengan detailFragment serta menambahkannya ke back stack. Terdapat Log.e untuk mencatat event saat tombol ini ditekan dan data yang dikirim.
    - Pada **baris 67 hingga 71**, RecyclerView (binding.rvCharacter) dikonfigurasi dengan LinearLayoutManager, characterAdapter yang sudah dibuat, dan setHasFixedSize(true) untuk optimasi karena ukuran item tidak berubah.
- Pada **baris 73 hingga 78**, didefinisikan fungsi observeCharacterList() yang menggunakan viewLifecycleOwner.lifecycleScope.launch untuk menjalankan coroutine yang akan mengamati viewModel.characterList (sebuah StateFlow). Ketika ada data baru, characterAdapter.setData(list) dipanggil untuk memperbarui RecyclerView. collectLatest digunakan untuk memastikan hanya data terbaru yang diproses.
- Pada **baris 80 hingga 83**, method onDestroyView() membersihkan _binding menjadi null untuk menghindari memory leaks.

5. **DetailFragment.kt:**

- **Pada baris 1 hingga 7**, dilakukan impor library yang diperlukan untuk membuat fragment dan menggunakan View Binding.
- **Pada baris 9 hingga 13**, didefinisikan kelas DetailFragment yang mewarisi Fragment. Di dalamnya, terdapat deklarasi variabel binding dengan pattern yang aman untuk memory leak.
- **Pada baris 15 hingga 32**, diimplementasikan method onCreateView() yang meng-inflate layout fragment_detail.xml menggunakan View Binding. **Pada baris 20-22**, data karakter diambil dari arguments. **Pada baris 24-28**, data tersebut ditampilkan pada view-view yang sesuai.
- **Pada baris 34 hingga 37**, diimplementasikan method onDestroyView() yang membersihkan referensi binding untuk mencegah memory leak.

6. **activity_main.xml:**

- **Pada baris 1 hingga 9**, didefinisikan layout utama aplikasi menggunakan FrameLayout. Layout ini diberi ID frame_container **pada baris 8**, yang digunakan sebagai container untuk fragment-fragment dalam aplikasi.

7. **fragment_home.xml:**

- **Pada baris 1 hingga 8**, didefinisikan layout untuk HomeFragment menggunakan ConstraintLayout. Layout ini dikonfigurasi dengan lebar dan tinggi wrap_content, yang mungkin perlu diubah menjadi match_parent untuk mengisi seluruh layar.
- **Pada baris 10 hingga 16**, didefinisikan RecyclerView dengan ID rv_character yang akan menampilkan daftar karakter. RecyclerView ini dikonfigurasi untuk mengisi seluruh parent layout menggunakan constraint.

8. **fragment_detail.xml:**

- **Pada baris 1 hingga 7**, didefinisikan layout untuk DetailFragment menggunakan ConstraintLayout dengan lebar dan tinggi match_parent.
- **Pada baris 9 hingga 16**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di tengah atas layout.
- **Pada baris 18 hingga 25**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini diposisikan di bawah gambar dan di tengah horizontal.
- **Pada baris 27 hingga 35**, didefinisikan TextView dengan ID tv_profile sebagai label untuk bagian profil. TextView ini diposisikan di bawah nama dengan alignment kiri dan style bold.
- **Pada baris 37 hingga 44**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini diposisikan di bawah label profil dengan alignment kiri.

9. **item_character.xml:**

- **Pada baris 1 hingga 6**, didefinisikan layout root untuk item karakter menggunakan ConstraintLayout dengan lebar match_parent dan tinggi wrap_content.
- **Pada baris 8 hingga 24**, didefinisikan CardView dengan ID card_view yang memberikan efek card dengan sudut melengkung dan bayangan. CardView ini dikonfigurasi dengan berbagai properti seperti margin, warna latar, radius sudut, dan elevasi.
- **Pada baris 26 hingga 30**, didefinisikan ConstraintLayout di dalam CardView yang akan menampung semua elemen UI item karakter.
- **Pada baris 32 hingga 42**, didefinisikan TextView dengan ID tv_character untuk menampilkan nama karakter. TextView ini dikonfigurasi dengan style bold, ukuran 20sp, dan warna teks hitam.

- **Pada baris 44 hingga 55**, didefinisikan ImageView dengan ID img_character untuk menampilkan gambar karakter. ImageView ini dikonfigurasi dengan ukuran tetap dan diposisikan di sebelah kiri layout.
- **Pada baris 57 hingga 69**, didefinisikan TextView dengan ID tv_deskripsi untuk menampilkan deskripsi karakter. TextView ini dikonfigurasi dengan ukuran 10sp dan warna teks hitam.
- **Pada baris 71 hingga 82**, didefinisikan Button dengan ID btn_detail untuk melihat detail karakter. Button ini dikonfigurasi dengan sudut melengkung dan warna teks hitam.
- **Pada baris 84 hingga 96**, didefinisikan Button dengan ID btn_link untuk membuka link karakter di browser. Button ini juga dikonfigurasi dengan sudut melengkung dan warna teks hitam.

10. HomeViewModel.kt:

- Pada **baris 1 hingga 11**, dilakukan impor library yang diperlukan seperti Resources, ViewModel, viewModelScope untuk coroutines, kelas Character dari model, dan komponen Flow dari kotlinx.coroutines.flow.
- Pada **baris 13**, _characterList dideklarasikan sebagai MutableStateFlow yang privat, diinisialisasi dengan daftar kosong. Ini akan menyimpan state daftar karakter saat ini.
- Pada **baris 14**, characterList dideklarasikan sebagai StateFlow yang diekspos ke luar (bersifat read-only dari luar ViewModel). Ini adalah flow yang akan diamati oleh UI (dalam hal ini, HomeFragment).

- Pada **baris 16 hingga 29**, didefinisikan fungsi privat getCharacterFlow() yang mengembalikan Flow<List<Character>>. Fungsi ini bertugas untuk mengambil data karakter dari resources.
    - Pada **baris 18 hingga 21**, data nama, tautan, deskripsi, dan ID drawable foto karakter diambil dari arrays yang didefinisikan di strings.xml dan arrays.xml menggunakan resources.
    - Pada **baris 23 hingga 26**, sebuah ArrayList dari Character dibuat. Setiap karakter diinisialisasi dengan data yang telah diambil, dan kemudian ditambahkan ke listCharacter.
    - Pada **baris 27**, dataPhoto.recycle() dipanggil untuk melepaskan resources yang terkait dengan TypedArray setelah tidak lagi digunakan.
    - Pada baris 28, listCharacter di-emit melalui flow.
- Pada **baris 31 hingga 39**, didefinisikan fungsi loadCharacters(). Fungsi ini yang akan dipanggil dari HomeFragment untuk memicu pemuatan data.
    - viewModelScope.launch digunakan untuk menjalankan coroutine di dalam scope ViewModel.
    - getCharacterFlow() dipanggil untuk mendapatkan flow data.
    - .onStart { _characterList.value = emptyList() } digunakan untuk meng-emit daftar kosong ke _characterList segera sebelum flow mulai mengumpulkan data. Ini berguna untuk menampilkan state loading atau mengosongkan data sebelumnya.
    - .collect { characters -> _characterList.value = characters } mengumpulkan data yang di-emit dari getCharacterFlow() dan memperbarui _characterList.value dengan data karakter yang baru.

11. ViewModelFactory.kt:

- Pada **baris 1 hingga 5**, dilakukan impor library yang dibutuhkan, yaitu Resources, ViewModel, ViewModelProvider, dan HomeViewModel.

- Pada **baris 7**, kelas HomeViewModelFactory didefinisikan, yang menerima Resources sebagai parameter konstruktor. Kelas ini mengimplementasikan ViewModelProvider.Factory.
- Pada **baris 8 hingga 12**, method create() di-override. Method ini akan dipanggil oleh sistem ketika instance ViewModel dibutuhkan.
    - Pada **baris 9**, diperiksa apakah modelClass adalah turunan dari HomeViewModel::class.java.
    - Jika iya (pada **baris 11**), instance HomeViewModel dibuat dengan meneruskan resources yang diterima oleh factory, dan kemudian di-cast ke tipe T (dengan @Suppress("UNCHECKED_CAST") untuk menekan peringatan unchecked cast).
    - Jika tidak (pada **baris 14**), IllegalArgumentException dilempar karena factory ini hanya tahu cara membuat HomeViewModel.

12. Debugger:

Debugger adalah alat yang sangat penting dalam pengembangan perangkat lunak. Fungsinya adalah untuk membantu programmer menemukan dan memperbaiki bug atau kesalahan dalam kode aplikasi. Dengan debugger, Anda dapat menjalankan aplikasi secara terkontrol, baris per baris kode, memeriksa nilai variabel pada waktu tertentu, dan memahami alur eksekusi program. Ini memungkinkan Anda untuk mengidentifikasi mengapa aplikasi tidak berjalan sesuai harapan.

Cara Menggunakan Debugger

Berikut adalah langkah-langkah umum cara menggunakan Debugger di Android Studio:

1) **Menetapkan Breakpoint:**
    - Breakpoint adalah titik dalam kode di mana eksekusi program akan berhenti sementara, memungkinkan Anda untuk memeriksa state aplikasi.

- Untuk menetapkan breakpoint, klik pada gutter (area di sebelah kiri nomor baris) di editor kode Android Studio pada baris yang ingin Anda selidiki. Sebuah titik merah akan muncul, menandakan breakpoint aktif. Anda perlu mencari setidaknya satu breakpoint yang relevan dengan aplikasi Anda.

2) **Memulai Sesi Debugging:**
   - Setelah breakpoint ditetapkan, jalankan aplikasi dalam mode debug. Anda bisa melakukannya dengan mengklik ikon "Debug 'app'" (biasanya bergambar serangga/bug) di toolbar Android Studio, atau melalui menu Run > Debug 'app'.
3) **Mengontrol Eksekusi:**
   - Ketika eksekusi program mencapai breakpoint, aplikasi akan berhenti, dan panel Debugger akan muncul di Android Studio. Panel ini menampilkan informasi seperti call stack (urutan pemanggilan fungsi), variabel lokal, dan nilai-nilainya.
   - Dari sini, Anda bisa menggunakan berbagai tombol kontrol untuk melanjutkan eksekusi.
4) **Memeriksa Variabel:**
   - Di panel Debugger, Anda dapat melihat nilai dari variabel-variabel yang ada dalam scope saat ini. Anda juga bisa "mengawasi" (watch) variabel tertentu untuk melihat perubahannya seiring eksekusi kode.

Fitur Step Into, Step Over, dan Step Out

Ketika eksekusi berhenti di sebuah breakpoint, Anda memiliki beberapa opsi untuk melanjutkan eksekusi baris per baris:

1) **Step Over:**
   - **Fungsi**: Mengeksekusi baris kode saat ini dan berpindah ke baris kode berikutnya dalam file yang sama.

- **Kapan digunakan**: Jika baris saat ini berisi pemanggilan sebuah fungsi, "Step Over" akan mengeksekusi seluruh fungsi tersebut tanpa masuk ke dalamnya, lalu berhenti di baris berikutnya setelah pemanggilan fungsi itu selesai. Ini berguna jika Anda yakin fungsi tersebut bekerja dengan benar dan tidak perlu memeriksanya secara detail.

2) **Step Into:**
   - **Fungsi**: Jika baris kode saat ini adalah pemanggilan sebuah fungsi, "Step Into" akan masuk ke dalam fungsi tersebut, memungkinkan Anda untuk men-debug baris per baris di dalam fungsi itu.
   - **Kapan digunakan**: Gunakan ini jika Anda ingin memeriksa bagaimana sebuah fungsi tertentu dieksekusi atau jika Anda mencurigai ada bug di dalam fungsi tersebut. Jika baris saat ini bukan pemanggilan fungsi, "Step Into" akan berperilaku sama seperti "Step Over".
3) **Step Out:**
   - **Fungsi**: Mengeksekusi sisa baris kode dalam fungsi saat ini dan kemudian berhenti pada baris kode berikutnya setelah fungsi tersebut selesai dieksekusi (yaitu, kembali ke kode yang memanggil fungsi tersebut).
   - **Kapan digunakan**: Jika Anda sudah masuk ke dalam sebuah fungsi menggunakan "Step Into" dan telah selesai memeriksa apa yang Anda butuhkan di sana, Anda dapat menggunakan "Step Out" untuk keluar dari fungsi tersebut dengan cepat tanpa harus menjalankan sisa barisnya satu per satu.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Jawaban

Dalam arsitektur aplikasi Android yang kompleks, Application class berperan sebagai komponen fundamental yang seringkali potensinya belum dimanfaatkan sepenuhnya. Ini adalah kelas dasar dalam sebuah aplikasi Android yang diinstansiasi sebelum komponen lain seperti Activity, Service, atau BroadcastReceiver ketika proses aplikasi dibuat. Anggaplah ini sebagai titik masuk paling awal untuk kode Anda, menyediakan sebuah singleton instance yang hidup selama proses aplikasi Anda berjalan. Memahami peran dan fungsinya adalah kunci untuk membangun aplikasi Android yang terstruktur dengan baik dan efisien.

Fungsi utama dari Application class adalah untuk memelihara state aplikasi global. Karena ia diinstansiasi pertama kali dan bertahan sepanjang siklus hidup aplikasi (kecuali prosesnya dihentikan), ini adalah tempat yang sesuai untuk menyimpan data atau objek yang perlu diakses secara luas di seluruh aplikasi. Ini bisa berkisar dari melakukan caching data bersama, mengelola sesi pengguna, atau menyimpan referensi ke kelas utilitas atau layanan yang dibutuhkan di berbagai bagian aplikasi.

Peran signifikan lainnya adalah inisialisasi global. Seringkali ada tugas atau library pihak ketiga yang perlu disiapkan sekali ketika aplikasi dimulai. Metode onCreate() dari Application class kustom adalah tempat yang sempurna untuk rutinitas inisialisasi sekali jalan tersebut. Ini bisa termasuk menyiapkan tools analitik, menginisialisasi framework logging, atau mengonfigurasi library jaringan. Dengan memusatkan inisialisasi ini di sini, Anda memastikan bahwa mereka hanya dilakukan sekali, mencegah panggilan setup yang berlebihan dan potensi konflik.

Lebih lanjut, Application class dapat merespons peristiwa siklus hidup tingkat aplikasi. Meskipun Activity memiliki siklus hidupnya sendiri yang detail, Application class juga memiliki callback seperti onCreate() dan onTerminate(). Meskipun onTerminate() tidak dijamin akan dipanggil pada perangkat produksi (ini terutama untuk lingkungan yang diemulasi), onCreate() dapat diandalkan untuk dipanggil dan merupakan kait utama untuk logika startup aplikasi. Ini juga dapat digunakan untuk bereaksi terhadap situasi memori rendah melalui onLowMemory() atau perubahan konfigurasi yang memengaruhi seluruh aplikasi melalui onConfigurationChanged(), meskipun penanganan yang terakhir lebih umum dilakukan di tingkat Activity.

Untuk memanfaatkan kemampuan ini, pengembang biasanya membuat subkelas kustom dari android.app.Application. Kelas kustom ini kemudian perlu dideklarasikan dalam file AndroidManifest.xml di dalam tag <application> menggunakan atribut android:name. Misalnya, jika Anda membuat kelas bernama MyGlobalApp, manifes Anda akan menyertakan <application android:name=".MyGlobalApp" ... >. Ini memberitahu sistem Android untuk menggunakan kelas kustom Anda sebagai titik masuk aplikasi.

Namun, sangat penting untuk menggunakan Application class dengan bijaksana. Membebaninya dengan terlalu banyak tanggung jawab atau menyimpan data dalam jumlah besar dapat menyebabkan desain monolitik dan peningkatan jejak memori. Ini juga dapat membuat pengujian menjadi lebih kompleks karena mengikat komponen dengan erat ke state global ini. Untuk mengelola dependensi dan state, pengembangan Android modern sering kali condong ke pola arsitektur seperti Dependency Injection (misalnya, Hilt atau Koin) yang dapat menawarkan solusi yang lebih skalabel dan dapat diuji daripada hanya mengandalkan Application class untuk semuanya.

Kesimpulannya, Application class adalah komponen yang kuat dalam framework Android, terutama dirancang untuk mengelola state aplikasi global dan melakukan inisialisasi sekali jalan. Ketika digunakan dengan cermat, ia menyediakan cara yang terpusat dan nyaman untuk menangani sumber daya dan konfigurasi yang menjangkau seluruh aplikasi. Namun, pengembang harus menyadari potensi kekurangannya dan menganggapnya sebagai salah satu alat di antara banyak alat lainnya dalam menyusun arsitektur aplikasi Android yang kuat dan dapat dipelihara.

MODUL 5 :
Connect to the Internet
SOAL 1

Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:

a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil

   data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.

b. Gunakan KotlinX Serialization sebagai library JSON.

c. Gunakan library seperti Coil atau Glide untuk image loading.

d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API:

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

e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)

f. Gunakan caching strategy pada Room..

g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

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

A. Source Code

1. MainActivity.kt

Tabel 25. Source Code MainActivity.kt

```
1  package com.example.monsterhunterarmor
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
```

```
5   import
    com.example.monsterhunterarmor.databinding.ActivityMainB
    inding
6
7   class MainActivity : AppCompatActivity() {
8       private lateinit var binding: ActivityMainBinding
9
10      override fun onCreate(savedInstanceState: Bundle?) {
11          super.onCreate(savedInstanceState)
12          binding =
    ActivityMainBinding.inflate(layoutInflater)
13          setContentView(binding.root)
14      }
15  }
```

2. ArmorAdapter.kt

Tabel 26. Source Code ArmorAdapter.kt

```
1   package com.example.monsterhunterarmor.adapter
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.DiffUtil
6   import androidx.recyclerview.widget.ListAdapter
7   import androidx.recyclerview.widget.RecyclerView
8   import com.bumptech.glide.Glide
9   import com.example.monsterhunterarmor.R
10  import
    com.example.monsterhunterarmor.data.local.ArmorEntity
11  import
    com.example.monsterhunterarmor.databinding.ItemArmorBind
    ing
12
13  class ArmorAdapter(
14      private val listener: OnArmorClickListener
15  ) : ListAdapter<ArmorEntity,
    ArmorAdapter.ArmorViewHolder>(DIFF_CALLBACK) {
16
17      interface OnArmorClickListener {
18          fun onDetailClick(armor: ArmorEntity)
19          fun onSearchClick(armor: ArmorEntity)
20      }
21
22      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): ArmorViewHolder {
23          val binding =
    ItemArmorBinding.inflate(LayoutInflater.from(parent.cont
    ext), parent, false)
```

```
24          return ArmorViewHolder(binding, listener)
25      }
26
27      override fun onBindViewHolder(holder:
    ArmorViewHolder, position: Int) {
28          val armor = getItem(position)
29          holder.bind(armor)
30      }
31
32      class ArmorViewHolder(
33          private val binding: ItemArmorBinding,
34          private val listener: OnArmorClickListener
35      ) : RecyclerView.ViewHolder(binding.root) {
36
37          fun bind(armor: ArmorEntity) {
38              binding.tvArmorName.text = armor.name
39              binding.tvArmorInfo.text = "Rank:
    ${armor.rank} | Type: ${armor.type}"
40
41              Glide.with(itemView.context)
42                  .load(armor.imageUrl)
43
    .placeholder(R.drawable.ic_launcher_background)
44
    .error(R.drawable.ic_launcher_foreground)
45                  .into(binding.ivArmorPhoto)
46
47              binding.btnDetail.setOnClickListener {
48                  listener.onDetailClick(armor)
49              }
50
51              binding.btnSearch.setOnClickListener {
52                  listener.onSearchClick(armor)
53              }
54          }
55      }
56
57      companion object {
58          val DIFF_CALLBACK = object :
    DiffUtil.ItemCallback<ArmorEntity>() {
59              override fun areItemsTheSame(oldItem:
    ArmorEntity, newItem: ArmorEntity): Boolean {
60                  return oldItem.id == newItem.id
61              }
62
63              override fun areContentsTheSame(oldItem:
    ArmorEntity, newItem: ArmorEntity): Boolean {
64                  return oldItem == newItem
65              }
```

```
66          }
67      }
68  }
```

3. ArmorDao.kt

Tabel 27. Source Code ArmorDao.kt

```
1   package com.example.monsterhunterarmor.data.local
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 ArmorDao {
11      @Query("SELECT * FROM armor")
12      fun getAllArmor(): Flow<List<ArmorEntity>>
13
14      @Insert(onConflict = OnConflictStrategy.REPLACE)
15      suspend fun insertAll(armorList: List<ArmorEntity>)
16
17      @Query("DELETE FROM armor")
18      suspend fun deleteAll()
19  }
```

4. ArmorDatabase.kt

Tabel 28. Source Code ArmorDatabase.kt

```
1   package com.example.monsterhunterarmor.data.local
2
3   import android.content.Context
4   import androidx.room.Database
5   import androidx.room.Room
6   import androidx.room.RoomDatabase
7   import androidx.room.TypeConverters
8
9   @Database(entities = [ArmorEntity::class], version = 2,
    exportSchema = false)
10  @TypeConverters(Converters::class)
11  abstract class ArmorDatabase : RoomDatabase() {
12      abstract fun armorDao(): ArmorDao
13
14      companion object {
15          @Volatile
16          private var INSTANCE: ArmorDatabase? = null
```

```
17
18          fun getDatabase(context: Context):
    ArmorDatabase {
19              return INSTANCE ?: synchronized(this) {
20                  val instance = Room.databaseBuilder(
21                      context.applicationContext,
22                      ArmorDatabase::class.java,
23                      "armor_database"
24                  )
25                      .fallbackToDestructiveMigration()
26                      .build()
27                  INSTANCE = instance
28                  instance
29              }
30          }
31      }
32  }
```

5. ArmorEntity.kt

Tabel 29. Source Code ArmorEntity.kt

```
1   package com.example.monsterhunterarmor.data.local
2
3   import android.os.Parcelable
4   import androidx.room.Embedded
5   import androidx.room.Entity
6   import androidx.room.PrimaryKey
7   import
    com.example.monsterhunterarmor.data.model.ArmorDefense
8   import
    com.example.monsterhunterarmor.data.model.ArmorResistanc
    es
9   import
    com.example.monsterhunterarmor.data.model.ArmorSkill
10  import kotlinx.parcelize.Parcelize
11
12  @Parcelize
13  @Entity(tableName = "armor")
14  data class ArmorEntity(
15      @PrimaryKey
16      val id: Int,
17      val name: String,
18      val rank: String,
19      val type: String,
20      val imageUrl: String?,
21
22      @Embedded
23      val defense: ArmorDefense,
```

```
24
25      @Embedded
26      val resistances: ArmorResistances,
27
28      val skills: List<ArmorSkill>
29
30  ) : Parcelable
```

6. Converters.kt

Tabel 30. Source Code Converters.kt

```
1   package com.example.monsterhunterarmor.data.local
2
3   import androidx.room.TypeConverter
4   import
    com.example.monsterhunterarmor.data.model.ArmorSkill
5   import kotlinx.serialization.encodeToString
6   import kotlinx.serialization.json.Json
7
8   class Converters {
9       @TypeConverter
10      fun fromSkillList(skills: List<ArmorSkill>): String
    {
11          return Json.encodeToString(skills)
12      }
13
14      @TypeConverter
15      fun toSkillList(skillsJson: String):
    List<ArmorSkill> {
16          return Json.decodeFromString(skillsJson)
17      }
18  }
```

7. ArmorModels.kt

Tabel 31. Source Code ArmorModels.kt

```
1   package com.example.monsterhunterarmor.data.model
2
3   import android.os.Parcelable
4   import kotlinx.parcelize.Parcelize
5   import kotlinx.serialization.Serializable
6
7   @Serializable
8   data class ArmorResponse(
9       val id: Int,
10      val name: String,
11      val rank: String,
```

```
12      val type: String,
13      val assets: ArmorAssets? = null,
14      val defense: ArmorDefense,
15      val resistances: ArmorResistances,
16      val skills: List<ArmorSkill>,
17      val slots: List<ArmorSlot>
18  )
19
20  @Serializable
21  data class ArmorAssets(
22      val imageMale: String? = null,
23      val imageFemale: String? = null
24  )
25
26  @Parcelize
27  @Serializable
28  data class ArmorDefense(
29      val base: Int,
30      val max: Int,
31      val augmented: Int
32  ) : Parcelable
33
34  @Parcelize
35  @Serializable
36  data class ArmorResistances(
37      val fire: Int,
38      val water: Int,
39      val ice: Int,
40      val thunder: Int,
41      val dragon: Int
42  ) : Parcelable
43
44  @Parcelize
45  @Serializable
46  data class ArmorSkill(
47      val id: Int,
48      val level: Int,
49      val skillName: String,
50      val description: String
51  ) : Parcelable
52
53  @Serializable
54  data class ArmorSlot(
55      val rank: Int
56  )
```

8. ApiResponse.kt

Tabel 32. Source Code ApiResponse.kt

```
1  package com.example.monsterhunterarmor.data.remote
2
3  sealed class ApiResponse<out R> {
4      data class Success<out T>(val data: T) :
   ApiResponse<T>()
5      data class Error(val errorMessage: String) :
   ApiResponse<Nothing>()
6      object Loading : ApiResponse<Nothing>()
7  }
```

9. ArmorApiService.kt

Tabel 33. Source Code ArmorApiService.kt

```
1  package com.example.monsterhunterarmor.data.remote
2
3  import
   com.example.monsterhunterarmor.data.model.ArmorResponse
4  import retrofit2.http.GET
5
6  interface ArmorApiService {
7      @GET("armor")
8      suspend fun getArmor(): List<ArmorResponse>
9  }
```

10. AppContainer.kt

Tabel 34. Source Code AppContainer.kt

```
1  package com.example.monsterhunterarmor.di
2
3  import android.content.Context
4  import
   com.example.monsterhunterarmor.data.local.ArmorDatabase
5  import
   com.example.monsterhunterarmor.data.remote.ArmorApiServi
   ce
6  import
   com.example.monsterhunterarmor.repository.ArmorReposito
   ry
7  import
   com.jakewharton.retrofit2.converter.kotlinx.serializatio
   n.asConverterFactory
```

```
8   import kotlinx.serialization.json.Json
9   import okhttp3.MediaType.Companion.toMediaType
10  import okhttp3.OkHttpClient
11  import okhttp3.logging.HttpLoggingInterceptor
12  import retrofit2.Retrofit
13
14  class AppContainer(private val context: Context) {
15
16      private val json = Json { ignoreUnknownKeys = true }
17
18      private val loggingInterceptor =
19
    HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor
    .Level.BODY)
20
21      private val client = OkHttpClient.Builder()
22          .addInterceptor(loggingInterceptor)
23          .build()
24
25      private val retrofit = Retrofit.Builder()
26          .baseUrl("https://mhw-db.com/")
27
    .addConverterFactory(json.asConverterFactory("applicatio
    n/json".toMediaType()))
28          .client(client)
29          .build()
30
31      private val apiService: ArmorApiService by lazy {
32          retrofit.create(ArmorApiService::class.java)
33      }
34
35      private val armorDb: ArmorDatabase by lazy {
36          ArmorDatabase.getDatabase(context)
37      }
38
39      val armorRepository: ArmorRepository by lazy {
40          ArmorRepository(apiService, armorDb.armorDao())
41      }
42  }
```

11. ArmorDetailFragment.kt

Tabel 35. Source Code ArmorDetailFragment.kt

```
1   package
    com.example.monsterhunterarmor.presentation.detail
2
3   import android.os.Bundle
4   import android.view.LayoutInflater
```

```
5  import android.view.View
6  import android.view.ViewGroup
7  import android.widget.TextView
8  import androidx.fragment.app.Fragment
9  import androidx.navigation.fragment.navArgs
10 import com.bumptech.glide.Glide
11 import
   com.example.monsterhunterarmor.data.model.ArmorSkill
12 import
   com.example.monsterhunterarmor.databinding.FragmentArmor
   DetailBinding
13
14 class ArmorDetailFragment : Fragment() {
15
16     private var _binding: FragmentArmorDetailBinding? =
   null
17     private val binding get() = _binding!!
18
19     private   val   args:   ArmorDetailFragmentArgs   by
   navArgs()
20
21     override fun onCreateView(
22         inflater: LayoutInflater, container: ViewGroup?,
23         savedInstanceState: Bundle?
24     ): View {
25         _binding                                        =
   FragmentArmorDetailBinding.inflate(inflater,   container,
   false)
26         return binding.root
27     }
28
29     override      fun      onViewCreated(view:      View,
   savedInstanceState: Bundle?) {
30         super.onViewCreated(view, savedInstanceState)
31         val armor = args.armor
32
33         Glide.with(this)
34             .load(armor.imageUrl)
35             .into(binding.ivArmorDetailPhoto)
36
37         binding.tvArmorDetailName.text = armor.name
38
39         val defense = armor.defense
40         binding.tvDefenseStats.text       =       "Base:
   ${defense.base}   |   Max:   ${defense.max}   |   Augmented:
   ${defense.augmented}"
41
42         val res = armor.resistances
```

```
43          binding.tvResistancesStats.text     =     "Fire:
    ${res.fire},      Water:      ${res.water},      Thunder:
    ${res.thunder}, Ice: ${res.ice}, Dragon: ${res.dragon}"
44
45          addSkillsToLayout(armor.skills)
46      }
47
48      private         fun         addSkillsToLayout(skills:
    List<ArmorSkill>) {
49          if (skills.isEmpty()) {
50              val noSkillsText = TextView(context).apply {
51                  text = "No skills available."
52
    setTextAppearance(com.google.android.material.R.style.Te
    xtAppearance_MaterialComponents_Body2)
53              }
54
    binding.llSkillsContainer.addView(noSkillsText)
55              return
56          }
57
58          skills.forEach { skill ->
59              val skillTitle = TextView(context).apply {
60                  text    =    "${skill.skillName}    (Lv.
    ${skill.level})"
61
    setTextAppearance(com.google.android.material.R.style.Te
    xtAppearance_MaterialComponents_Subtitle1)
62                  layoutParams                            =
    ViewGroup.MarginLayoutParams(
63                      ViewGroup.LayoutParams.MATCH_PARENT,
64                      ViewGroup.LayoutParams.WRAP_CONTENT
65                  ).apply {
66                      topMargin            =            if
    (binding.llSkillsContainer.childCount > 1) 24 else 8
67                  }
68              }
69
70              val            skillDescription            =
    TextView(context).apply {
71                  text = skill.description
72
    setTextAppearance(com.google.android.material.R.style.Te
    xtAppearance_MaterialComponents_Body2)
73              }
74
75
    binding.llSkillsContainer.addView(skillTitle)
```

```
76
    binding.llSkillsContainer.addView(skillDescription)
77          }
78      }
79
80      override fun onDestroyView() {
81          super.onDestroyView()
82          _binding = null
83      }
84  }
```

12. ArmorListFragment.kt

Tabel 36. Source Code ArmorListFragment.kt

```
1   package
    com.example.monsterhunterarmor.presentation.home
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 android.widget.Toast
11  import androidx.core.view.isVisible
12  import androidx.fragment.app.Fragment
13  import androidx.fragment.app.viewModels
14  import androidx.lifecycle.Lifecycle
15  import androidx.lifecycle.lifecycleScope
16  import androidx.lifecycle.repeatOnLifecycle
17  import androidx.navigation.fragment.findNavController
18  import androidx.recyclerview.widget.LinearLayoutManager
19  import com.example.monsterhunterarmor.ArmorApplication
20  import
    com.example.monsterhunterarmor.adapter.ArmorAdapter
21  import
    com.example.monsterhunterarmor.data.local.ArmorEntity
22  import
    com.example.monsterhunterarmor.data.remote.ApiResponse
23  import
    com.example.monsterhunterarmor.databinding.FragmentArmo
    rListBinding
24  import
    com.example.monsterhunterarmor.utils.ViewModelFactory
25  import kotlinx.coroutines.launch
26
27  class ArmorListFragment : Fragment() {
```

```
28
29      private var _binding: FragmentArmorListBinding? =
    null
30      private val binding get() = _binding!!
31
32      private val viewModel: ArmorViewModel by viewModels
    {
33          ViewModelFactory((requireActivity().application
    as ArmorApplication).appContainer.armorRepository)
34      }
35
36      private val armorAdapter = ArmorAdapter(object :
    ArmorAdapter.OnArmorClickListener {
37          override fun onDetailClick(armor: ArmorEntity) {
38              viewModel.onDetailButtonClicked(armor)
39          }
40
41          override fun onSearchClick(armor: ArmorEntity) {
42              viewModel.onSearchButtonClicked(armor)
43          }
44      })
45
46      override fun onCreateView(
47          inflater: LayoutInflater, container: ViewGroup?,
48          savedInstanceState: Bundle?
49      ): View {
50          _binding                                        =
    FragmentArmorListBinding.inflate(inflater,   container,
    false)
51          return binding.root
52      }
53
54      override     fun     onViewCreated(view:     View,
    savedInstanceState: Bundle?) {
55          super.onViewCreated(view, savedInstanceState)
56
57          setupRecyclerView()
58          observeArmorData()
59          observeViewEvents()
60      }
61
62      private fun setupRecyclerView() {
63          binding.rvArmor.apply {
64              adapter = armorAdapter
65              layoutManager = LinearLayoutManager(context)
66          }
67      }
68
69      private fun observeArmorData() {
```

```
70          viewLifecycleOwner.lifecycleScope.launch {
71              repeatOnLifecycle(Lifecycle.State.STARTED)
    {
72                  viewModel.armorState.collect { response
    ->
73                      when (response) {
74                          is ApiResponse.Loading -> {
75
    binding.progressBar.isVisible = true
76                          }
77                          is ApiResponse.Success -> {
78                              response.data.collect      {
    armorList ->
79
    binding.progressBar.isVisible = false
80
    armorAdapter.submitList(armorList)
81                                  if
    (armorList.isNotEmpty()) {
82
    Log.d("ArmorListFragment",    "${armorList.size}    items
    submitted.")
83                                  }
84                              }
85                          }
86                          is ApiResponse.Error -> {
87
    binding.progressBar.isVisible = false
88                              Toast.makeText(context,
    response.errorMessage, Toast.LENGTH_LONG).show()
89                          }
90                      }
91                  }
92              }
93          }
94      }
95
96      private fun observeViewEvents() {
97          viewLifecycleOwner.lifecycleScope.launch {
98              repeatOnLifecycle(Lifecycle.State.STARTED)
    {
99                  viewModel.eventFlow.collect { event ->
100                     when (event) {
101                         is
    ArmorViewModel.ViewEvent.NavigateToDetail -> {
102                             Log.d("ArmorListFragment",
    "Navigating  to  detail  for:  ${event.armor.name}  (ID:
    ${event.armor.id})")
```

```
103                             val         action         =
    ArmorListFragmentDirections.actionArmorListFragmentToAr
    morDetailFragment(event.armor)
104
    findNavController().navigate(action)
105                         }
106                         is
    ArmorViewModel.ViewEvent.OpenBrowser -> {
107                             val         query          =
    "https://monsterhunterworld.wiki.fextralife.com/${event
    .query.replace(" ", "+")}"
108                             val         intent         =
    Intent(Intent.ACTION_VIEW, Uri.parse(query))
109                             startActivity(intent)
110                         }
111                     }
112                 }
113             }
114         }
115     }
116
117     override fun onDestroyView() {
118         super.onDestroyView()
119         binding.rvArmor.adapter = null
120         _binding = null
121     }
122 }
```

13. ArmorListFragment.kt

Tabel 37. Source Code ArmorViewModel.kt

```
1   package com.example.monsterhunterarmor.presentation.home
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import
    com.example.monsterhunterarmor.data.local.ArmorEntity
7   import
    com.example.monsterhunterarmor.data.remote.ApiResponse
8   import
    com.example.monsterhunterarmor.repository.ArmorRepositor
    y
9   import kotlinx.coroutines.flow.Flow
10  import kotlinx.coroutines.flow.MutableSharedFlow
11  import kotlinx.coroutines.flow.MutableStateFlow
12  import kotlinx.coroutines.flow.asSharedFlow
13  import kotlinx.coroutines.flow.StateFlow
```

```
14 import kotlinx.coroutines.flow.collectLatest
15 import kotlinx.coroutines.launch
16
17 class ArmorViewModel(private val repository:
   ArmorRepository) : ViewModel() {
18
19     private val _armorState =
   MutableStateFlow<ApiResponse<Flow<List<ArmorEntity>>>>(A
   piResponse.Loading)
20     val armorState:
   StateFlow<ApiResponse<Flow<List<ArmorEntity>>>> =
   _armorState
21
22     private val _eventFlow =
   MutableSharedFlow<ViewEvent>()
23     val eventFlow = _eventFlow.asSharedFlow()
24
25     init {
26         fetchArmor()
27     }
28
29     fun fetchArmor() {
30         viewModelScope.launch {
31             repository.getArmorList().collectLatest {
32                 _armorState.value = it
33             }
34         }
35     }
36
37     fun onDetailButtonClicked(armor: ArmorEntity) {
38         Log.d("ArmorViewModel", "Detail button clicked
   for: ${armor.name}")
39         viewModelScope.launch {
40
   _eventFlow.emit(ViewEvent.NavigateToDetail(armor))
41         }
42     }
43
44     fun onSearchButtonClicked(armor: ArmorEntity) {
45         Log.d("ArmorViewModel", "Search button clicked
   for: ${armor.name}")
46         viewModelScope.launch {
47
   _eventFlow.emit(ViewEvent.OpenBrowser(armor.name))
48         }
49     }
50
51     sealed class ViewEvent {
```

```
52          data    class    NavigateToDetail(val    armor:
    ArmorEntity) : ViewEvent()
53          data  class  OpenBrowser(val  query:  String)  :
    ViewEvent()
54      }
55  }
```

14. ArmorRepository.kt

Tabel 38. Source Code ArmorRepository.kt

```
1   package com.example.monsterhunterarmor.repository
2
3   import
    com.example.monsterhunterarmor.data.local.ArmorDao
4   import
    com.example.monsterhunterarmor.data.local.ArmorEntity
5   import
    com.example.monsterhunterarmor.data.remote.ApiResponse
6   import
    com.example.monsterhunterarmor.data.remote.ArmorApiServi
    ce
7   import kotlinx.coroutines.flow.Flow
8   import kotlinx.coroutines.flow.flow
9   import java.lang.Exception
10
11  class ArmorRepository(
12      private val apiService: ArmorApiService,
13      private val armorDao: ArmorDao
14  ) {
15      fun                                    getArmorList():
    Flow<ApiResponse<Flow<List<ArmorEntity>>>> = flow {
16          emit(ApiResponse.Loading)
17          val localData = armorDao.getAllArmor()
18          emit(ApiResponse.Success(localData))
19
20          try {
21              val response = apiService.getArmor()
22              val    armorEntities    =    response.map    {
    armorResponse ->
23                  ArmorEntity(
24                      id = armorResponse.id,
25                      name = armorResponse.name,
26                      rank = armorResponse.rank,
27                      type = armorResponse.type,
28                      imageUrl                              =
    armorResponse.assets?.imageMale                          ?:
    armorResponse.assets?.imageFemale,
29                      defense = armorResponse.defense,
```

```
30                      resistances                                 =
armorResponse.resistances,
31                      skills = armorResponse.skills
32                  )
33              }
34              armorDao.deleteAll()
35              armorDao.insertAll(armorEntities)
36          } catch (e: Exception) {
37              emit(ApiResponse.Error("Failed to fetch from
network: ${e.message}"))
38              e.printStackTrace()
39          }
40      }
41  }
```

15. ViewModelFactory.kt

Tabel 39. Source Code ViewModelFactory.kt

```
1   package com.example.monsterhunterarmor.utils
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5   import
    com.example.monsterhunterarmor.presentation.home.ArmorVi
    ewModel
6   import
    com.example.monsterhunterarmor.repository.ArmorRepositor
    y
7
8   class    ViewModelFactory(private    val    repository:
    ArmorRepository) : ViewModelProvider.NewInstanceFactory()
    {
9       @Suppress("UNCHECKED_CAST")
10      override  fun  <T  :  ViewModel>  create(modelClass:
    Class<T>): T {
11          if
    (modelClass.isAssignableFrom(ArmorViewModel::class.java)
    ) {
12              return ArmorViewModel(repository) as T
13          }
14          throw IllegalArgumentException("Unknown ViewModel
    class: " + modelClass.name)
15      }
16  }
```

16. ArmorApplication.kt

Tabel 40. Source Code ArmorApplication.kt

```
1  package com.example.monsterhunterarmor
2
3  import android.app.Application
4  import com.example.monsterhunterarmor.di.AppContainer
5
6  class ArmorApplication : Application() {
7      lateinit var appContainer: AppContainer
8      override fun onCreate() {
9          super.onCreate()
10         appContainer = AppContainer(this)
11     }
12 }
```

17. activity_main.xml

Tabel 41. Source Code activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.fragment.app.FragmentContainerView
   xmlns:android="http://schemas.android.com/apk/res/androi
   d"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:id="@+id/nav_host_fragment"
6
   android:name="androidx.navigation.fragment.NavHostFragme
   nt"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      app:defaultNavHost="true"
10     app:navGraph="@navigation/main_nav_graph"
11     tools:context=".MainActivity" />
```

18. fragment_armor_detail.xml

Tabel 42. Source Code fragment_armor_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
   xmlns:android="http://schemas.android.com/apk/res/andro
   id"
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=".presentation.detail.ArmorDetailFragment
        ">
8
9       <androidx.constraintlayout.widget.ConstraintLayout
10          android:layout_width="match_parent"
11          android:layout_height="wrap_content"
12          android:paddingBottom="16dp">
13
14          <ImageView
15              android:id="@+id/iv_armor_detail_photo"
16              android:layout_width="0dp"
17              android:layout_height="300dp"
18              android:scaleType="centerCrop"
19
        android:contentDescription="@string/armor_image"
20              app:layout_constraintEnd_toEndOf="parent"
21
        app:layout_constraintStart_toStartOf="parent"
22              app:layout_constraintTop_toTopOf="parent"
23
        tools:src="@tools:sample/backgrounds/scenic" />
24
25          <TextView
26              android:id="@+id/tv_armor_detail_name"
27              android:layout_width="0dp"
28              android:layout_height="wrap_content"
29              android:layout_marginHorizontal="16dp"
30              android:layout_marginTop="16dp"
31
        android:textAppearance="?attr/textAppearanceHeadlineSma
        ll"
32              android:textStyle="bold"
33              app:layout_constraintEnd_toEndOf="parent"
34
        app:layout_constraintStart_toStartOf="parent"
35
        app:layout_constraintTop_toBottomOf="@id/iv_armor_detai
        l_photo"
36              tools:text="Direwolf Armor" />
37
38
        <com.google.android.material.card.MaterialCardView
39              android:id="@+id/card_defense"
40              android:layout_width="0dp"
41              android:layout_height="wrap_content"
42              android:layout_marginTop="16dp"
```

```
43              app:cardCornerRadius="12dp"
44
    app:layout_constraintEnd_toEndOf="@+id/tv_armor_detail_
    name"
45
    app:layout_constraintStart_toStartOf="@+id/tv_armor_det
    ail_name"
46
    app:layout_constraintTop_toBottomOf="@+id/tv_armor_deta
    il_name">
47
48              <LinearLayout
49                  android:layout_width="match_parent"
50                  android:layout_height="wrap_content"
51                  android:orientation="vertical"
52                  android:padding="16dp">
53
54                  <TextView
55                      android:layout_width="wrap_content"
56
    android:layout_height="wrap_content"
57                      android:text="Defense"
58
    android:textAppearance="?attr/textAppearanceTitleMedium
    " />
59
60                  <TextView
61                      android:id="@+id/tv_defense_stats"
62                      android:layout_width="wrap_content"
63
    android:layout_height="wrap_content"
64                      android:layout_marginTop="8dp"
65
    android:textAppearance="?attr/textAppearanceBodyMedium"
66                      tools:text="Base: 70 | Max: 110 |
    Augmented: 140" />
67              </LinearLayout>
68
    </com.google.android.material.card.MaterialCardView>
69
70
    <com.google.android.material.card.MaterialCardView
71              android:id="@+id/card_resistances"
72              android:layout_width="0dp"
73              android:layout_height="wrap_content"
74              android:layout_marginTop="16dp"
75              app:cardCornerRadius="12dp"
76
    app:layout_constraintEnd_toEndOf="@+id/card_defense"
```

```
77
    app:layout_constraintStart_toStartOf="@+id/card_defense
    "
78
    app:layout_constraintTop_toBottomOf="@+id/card_defense"
    >
79
80              <LinearLayout
81                  android:layout_width="match_parent"
82                  android:layout_height="wrap_content"
83                  android:orientation="vertical"
84                  android:padding="16dp">
85
86                  <TextView
87                      android:layout_width="wrap_content"
88
    android:layout_height="wrap_content"
89                      android:text="Resistances"
90
    android:textAppearance="?attr/textAppearanceTitleMedium
    " />
91
92                  <TextView
93
    android:id="@+id/tv_resistances_stats"
94                      android:layout_width="wrap_content"
95
    android:layout_height="wrap_content"
96                      android:layout_marginTop="8dp"
97
    android:textAppearance="?attr/textAppearanceBodyMedium"
98                      tools:text="Fire:   2,   Water:   -1,
    Thunder: 0, Ice: 0, Dragon: 3" />
99              </LinearLayout>
100
    </com.google.android.material.card.MaterialCardView>
101
102
    <com.google.android.material.card.MaterialCardView
103             android:id="@+id/card_skills"
104             android:layout_width="0dp"
105             android:layout_height="wrap_content"
106             android:layout_marginTop="16dp"
107             app:cardCornerRadius="12dp"
108
    app:layout_constraintEnd_toEndOf="@+id/card_resistances
    "
```

```
109
    app:layout_constraintStart_toStartOf="@+id/card_resista
    nces"
110
    app:layout_constraintTop_toBottomOf="@+id/card_resistan
    ces">
111
112             <LinearLayout
113                 android:id="@+id/ll_skills_container"
114                 android:layout_width="match_parent"
115                 android:layout_height="wrap_content"
116                 android:orientation="vertical"
117                 android:padding="16dp">
118
119                 <TextView
120                     android:layout_width="wrap_content"
121
    android:layout_height="wrap_content"
122                     android:layout_marginBottom="8dp"
123                     android:text="Skills"
124
    android:textAppearance="?attr/textAppearanceTitleMedium
    " />
125             </LinearLayout>
126
    </com.google.android.material.card.MaterialCardView>
127
128
    </androidx.constraintlayout.widget.ConstraintLayout>
129 </ScrollView>
```

19. fragment_armor_list.xml

Tabel 43. Source Code fragment_armor_list.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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=".presentation.home.ArmorListFragment">
8
9       <androidx.recyclerview.widget.RecyclerView
10          android:id="@+id/rv_armor"
11          android:layout_width="0dp"
```

```
12          android:layout_height="0dp"
13          app:layout_constraintBottom_toBottomOf="parent"
14          app:layout_constraintEnd_toEndOf="parent"
15          app:layout_constraintStart_toStartOf="parent"
16          app:layout_constraintTop_toTopOf="parent"
17          tools:listitem="@layout/item_armor" />
18
19      <ProgressBar
20          android:id="@+id/progress_bar"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:visibility="gone"
24          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  </androidx.constraintlayout.widget.ConstraintLayout>
```

20. item_armor.xml

Tabel 44. Source Code item_armor.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <com.google.android.material.card.MaterialCardView
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
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="wrap_content"
7       android:layout_marginHorizontal="16dp"
8       android:layout_marginVertical="8dp"
9       app:cardCornerRadius="16dp"
10      app:cardElevation="4dp">
11
12      <androidx.constraintlayout.widget.ConstraintLayout
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:padding="12dp">
16
17
    <com.google.android.material.card.MaterialCardView
18              android:id="@+id/card_image"
19              android:layout_width="110dp"
20              android:layout_height="110dp"
21              app:cardCornerRadius="12dp"
22              app:cardElevation="0dp"
```

```
23
    app:layout_constraintStart_toStartOf="parent"
24              app:layout_constraintTop_toTopOf="parent">
25
26              <ImageView
27                  android:id="@+id/iv_armor_photo"
28                  android:layout_width="match_parent"
29                  android:layout_height="match_parent"
30
    android:contentDescription="@string/armor_image"
31                  android:scaleType="centerCrop"
32                  tools:src="@tools:sample/avatars" />
33
    </com.google.android.material.card.MaterialCardView>
34
35          <TextView
36              android:id="@+id/tv_armor_name"
37              android:layout_width="0dp"
38              android:layout_height="wrap_content"
39              android:layout_marginStart="16dp"
40              android:ellipsize="end"
41              android:maxLines="2"
42
    android:textAppearance="?attr/textAppearanceTitleMedium"
43              android:textStyle="bold"
44              app:layout_constraintEnd_toEndOf="parent"
45
    app:layout_constraintStart_toEndOf="@+id/card_image"
46              app:layout_constraintTop_toTopOf="parent"
47              tools:text="Direwolf Mail" />
48
49          <TextView
50              android:id="@+id/tv_armor_info"
51              android:layout_width="0dp"
52              android:layout_height="wrap_content"
53              android:layout_marginTop="4dp"
54
    android:textAppearance="?attr/textAppearanceBodySmall"
55
    app:layout_constraintEnd_toEndOf="@+id/tv_armor_name"
56
    app:layout_constraintStart_toStartOf="@+id/tv_armor_name
    "
57
    app:layout_constraintTop_toBottomOf="@+id/tv_armor_name"
58              tools:text="Rank: G | Type: Chest" />
59
60          <androidx.constraintlayout.widget.Barrier
61              android:id="@+id/content_barrier"
```

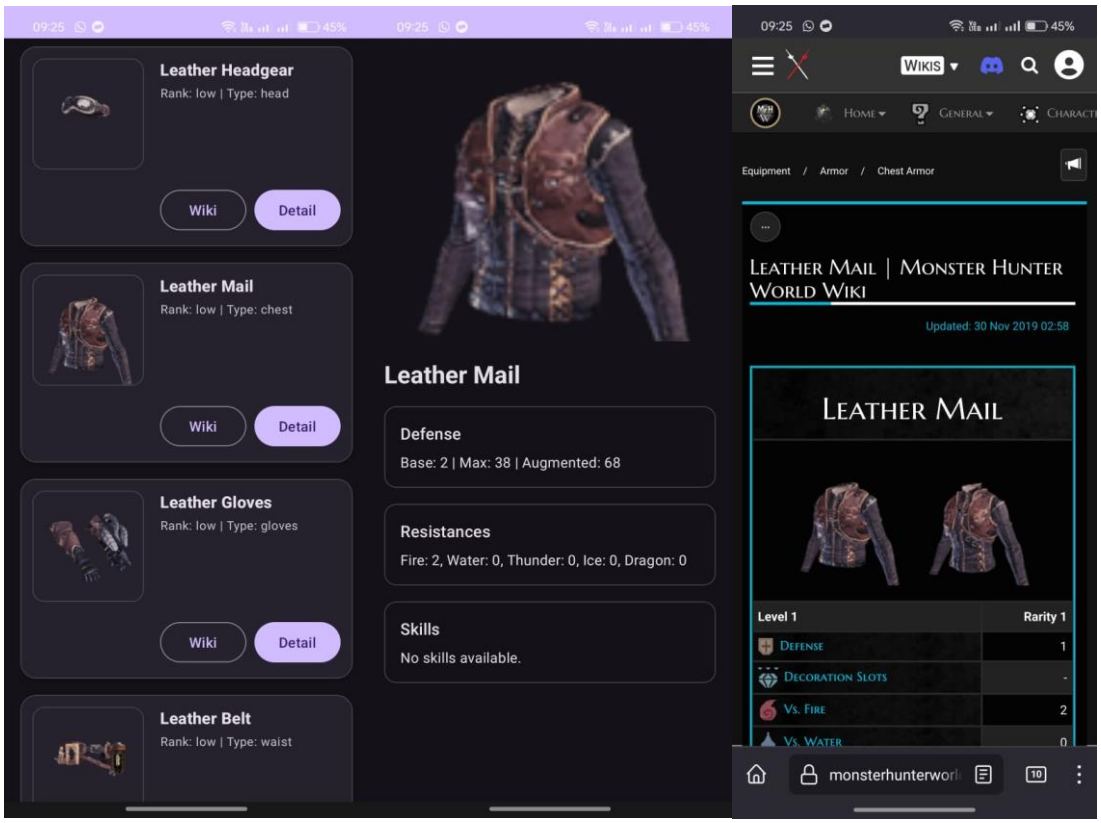
```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:barrierDirection="bottom"

app:constraint_referenced_ids="card_image,tv_armor_info"
/>


        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:orientation="horizontal"

app:layout_constraintEnd_toEndOf="@+id/tv_armor_name"

app:layout_constraintStart_toStartOf="@+id/tv_armor_name
"

app:layout_constraintTop_toBottomOf="@id/content_barrier
">


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

style="?attr/materialButtonOutlinedStyle"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginEnd="4dp"
                android:layout_weight="1"
                android:text="@string/search_on_web" />


<com.google.android.material.button.MaterialButton
                android:id="@+id/btn_detail"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginStart="4dp"
                android:layout_weight="1"
                android:text="@string/detail" />
        </LinearLayout>

    </androidx.constraintlayout.widget.ConstraintLayout>
</com.google.android.material.card.MaterialCardView>
```

21. main_nav_graph.xml

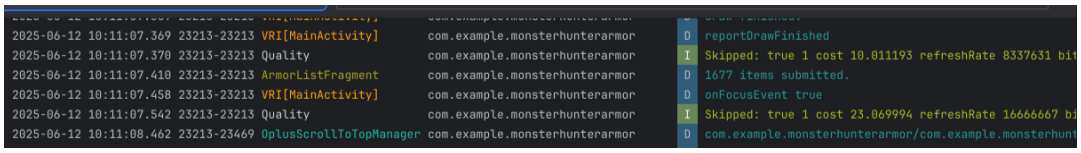
Tabel 45. Source Code main_nav_graph.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/androi
    d"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/main_nav_graph"
6       app:startDestination="@id/armorListFragment">
7
8       <fragment
9           android:id="@+id/armorListFragment"
10
    android:name="com.example.monsterhunterarmor.presentatio
    n.home.ArmorListFragment"
11          android:label="Armor List"
12          tools:layout="@layout/fragment_armor_list" >
13          <action
14
    android:id="@+id/action_armorListFragment_to_armorDetail
    Fragment"
15              app:destination="@id/armorDetailFragment" />
16      </fragment>
17
18      <fragment
19          android:id="@+id/armorDetailFragment"
20
    android:name="com.example.monsterhunterarmor.presentatio
    n.detail.ArmorDetailFragment"
21          android:label="Armor Detail"
22          tools:layout="@layout/fragment_armor_detail" >
23          <argument
24              android:name="armor"
25
    app:argType="com.example.monsterhunterarmor.data.local.A
    rmorEntity" />
26      </fragment>
27
28  </navigation>
```

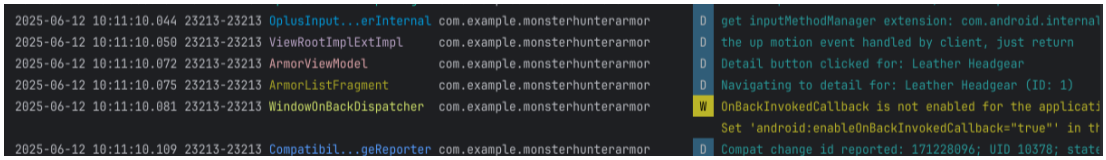
B. Output Program



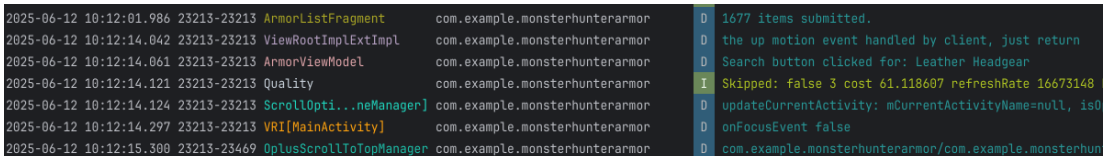
Gambar 15. Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 16. Screenshot Log Saat Data Item Masuk Ke Dalam List Modul 5



Gambar 17. Screenshot Log Saat Tombol Detail Dan Data Dari List Yang Dipilih Ketika Berpindah Ke Halaman Detail Modul 5



Gambar 18. Screenshot Log Tombol Explicit Intent Ditekan Modul 5

C. Pembahasan

1. MainActivity.kt:

Merupakan entry point atau titik masuk utama aplikasi. Kelas MainActivity berfungsi sebagai host untuk NavHostFragment, yang bertanggung jawab memuat dan menampilkan berbagai fragment sesuai dengan alur navigasi aplikasi yang didefinisikan dalam main_nav_graph.xml.

2. ArmorApplication.kt:

Kelas Application kustom yang diinisialisasi saat aplikasi pertama kali dijalankan. Peran utamanya adalah untuk membuat dan mengelola dependency injection container (AppContainer) sebagai singleton, memastikan dependensi seperti repository dan database tersedia untuk seluruh siklus hidup aplikasi.

3. adapter/ArmorAdapter.kt:

Sebuah RecyclerView.Adapter yang bertanggung jawab untuk mengikat data armor (List<ArmorEntity>) ke tampilan item dalam daftar. ArmorAdapter menangani pembuatan ViewHolder, pengisian data ke setiap item, dan pengelolaan interaksi pengguna seperti klik, yang kemudian meneruskan event tersebut untuk diproses lebih lanjut (misalnya, navigasi ke halaman detail).

4. data/local/ArmorEntity.kt:

Sebuah data class yang berfungsi sebagai model tabel untuk database Room. Kelas ini menggunakan anotasi @Entity untuk mendefinisikan tabel armor. Setiap properti dalam kelas ini merepresentasikan kolom di dalam tabel tersebut.

5. data/local/ArmorDao.kt:

DAO (Data Access Object). Interface ini berisi deklarasi fungsi-fungsi untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada tabel armor. Implementasi konkret dari DAO ini disediakan secara otomatis oleh Room.

6. **data/local/ArmorDatabase.kt:**

Kelas abstrak yang mewarisi RoomDatabase dan berfungsi sebagai konfigurasi utama untuk database aplikasi. Kelas ini mendefinisikan daftar entities (tabel) dan menyediakan akses ke DAO.

7. **data/local/Converters.kt:**

Menyediakan fungsi konversi tipe data (Type Converters) untuk Room. Fungsinya adalah mengubah tipe data kompleks yang tidak didukung secara native oleh Room (seperti List<ArmorSkill>) menjadi tipe data primitif (misalnya String JSON) agar dapat disimpan di database, dan sebaliknya.

8. **data/model/ArmorModels.kt:**

Berisi kumpulan data class yang merepresentasikan struktur data dari respons JSON API. Kelas-kelas ini digunakan oleh library kotlinx.serialization untuk melakukan parsing (deserialisasi) dari JSON menjadi objek Kotlin.

9. **data/remote/ArmorApiService.kt:**

Interface yang digunakan oleh Retrofit untuk mendefinisikan endpoints dari API. Setiap fungsi di dalam interface ini merepresentasikan satu panggilan API, seperti getArmorList() untuk mengambil daftar armor dari server.

10. **data/remote/ApiResponse.kt:**

Sebuah sealed class yang digunakan untuk membungkus respons dari panggilan jaringan. Ini memungkinkan penanganan state UI secara eksplisit untuk kondisi Loading, Success (dengan data), dan Error (dengan pesan kesalahan).

11. **di/AppContainer.kt:**

Berfungsi sebagai dependency injection container manual. Kelas ini bertanggung jawab untuk membuat dan menyediakan instance dari dependensi penting di seluruh aplikasi, seperti ArmorRepository dan ArmorApiService, untuk mempromosikan loose coupling dan kemudahan pengujian.

12. presentation/detail/ArmorDetailFragment.kt:

Fragment yang bertanggung jawab untuk menampilkan layar detail dari satu item armor. Fragment ini menerima data armor yang dipilih melalui Navigation Safe Args dan menampilkannya pada komponen UI yang sesuai.

13. presentation/home/ArmorListFragment.kt:

Fragment yang menjadi layar utama aplikasi. Bertugas menampilkan daftar armor dalam sebuah RecyclerView. Fragment ini mengobservasi data dari ArmorViewModel, menampilkan loading state dan pesan error, serta menangani input pengguna seperti pencarian dan navigasi.

14. presentation/home/ArmorViewModel.kt:

Komponen ViewModel yang menyediakan data dan state untuk ArmorListFragment. ArmorViewModel berinteraksi dengan ArmorRepository untuk mengambil data, mengelola state UI menggunakan StateFlow, dan menangani event dari UI (seperti klik) menggunakan Channel untuk memastikan logika terpisah dari tampilan.

15. repository/ArmorRepository.kt:

Bertindak sebagai Single Source of Truth (SSOT). Repository ini mengabstraksi sumber data (API atau database lokal) dari ViewModel. Ia berisi logika untuk mengambil data dari jaringan, menyimpannya ke dalam cache (database), dan menyediakan data yang konsisten untuk aplikasi.

16. utils/ViewModelFactory.kt:

Sebuah factory class yang bertujuan untuk membuat instance dari ArmorViewModel. Diperlukan karena ArmorViewModel memiliki dependensi (ArmorRepository) yang harus diinjeksikan saat pembuatan, sehingga tidak bisa menggunakan konstruktor default.

17. layout/activity_main.xml:

File layout untuk MainActivity. Berisi sebuah FragmentContainerView yang berfungsi sebagai wadah utama untuk NavHostFragment. Semua layar (fragment) dalam aplikasi akan dimuat di dalam container ini.

18. layout/fragment_armor_list.xml:

File layout XML yang mendefinisikan antarmuka pengguna untuk layar daftar armor. Layout ini berisi RecyclerView untuk menampilkan daftar data, ProgressBar sebagai indikator pemuatan, SearchView untuk fungsionalitas pencarian, dan TextView untuk menampilkan pesan status.

19. layout/fragment_armor_detail.xml:

Layout untuk halaman detail armor. Terdiri dari berbagai komponen seperti ImageView untuk gambar armor dan beberapa TextView untuk menampilkan atribut-atribut rinci seperti nama, rank, tipe, statistik pertahanan, dan resistensi.

20. layout/item_armor.xml:

Mendefinisikan layout untuk satu item dalam RecyclerView di ArmorListFragment. Layout ini berfungsi sebagai templat untuk setiap baris, yang menampilkan informasi ringkas seperti gambar, nama, dan tipe armor.

21. navigation/main_nav_graph.xml:

Pusat kendali navigasi aplikasi yang menggunakan Navigation Component. File ini mendefinisikan semua destinasi (fragment) dan actions (perpindahan antar fragment). Di sini, ArmorListFragment ditetapkan sebagai start destination dan didefinisikan pula aksi navigasi ke ArmorDetailFragment beserta argumen yang dikirimkan.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/PutraWhyra789/praktikum_pemrograman_mobile.git