LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Aiko Anatasha Wendiono **NIM. 2310817320013**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

JUNI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Aiko Anatasha Wendiono
NIM : 2310817320013

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Aulia Akbar	Muti'a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 198810272019032013

DAFTAR ISI

DAFTAR GAMBAR

MODUL 1: Android Basic with Kotlin

MODUL 2: Android Layout

MODUL 3: Build a Scrollable List

MODUL 4: ViewModel dan Debugging

MODUL 5: Connect to the Internet

DAFTAR TABEL

MODUL 1: Android Basic with Kotlin

MODUL 2: Android Layout

MODUL 3: Build a Scrollable List

MODUL 4: ViewModel dan Debugging

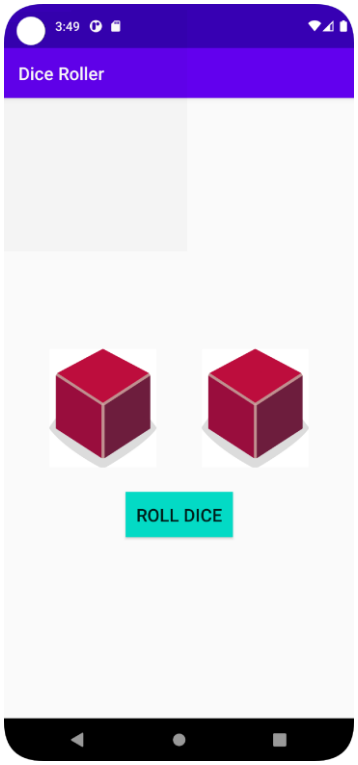
MODUL 5: Connect to the Internet

MODUL 1 : Android Basic with Kotlin

SOAL 1

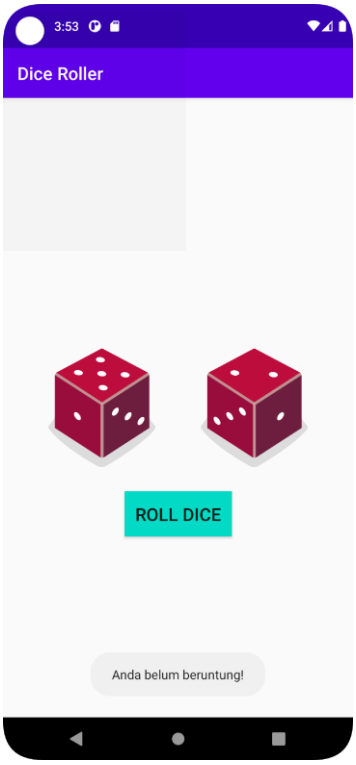
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



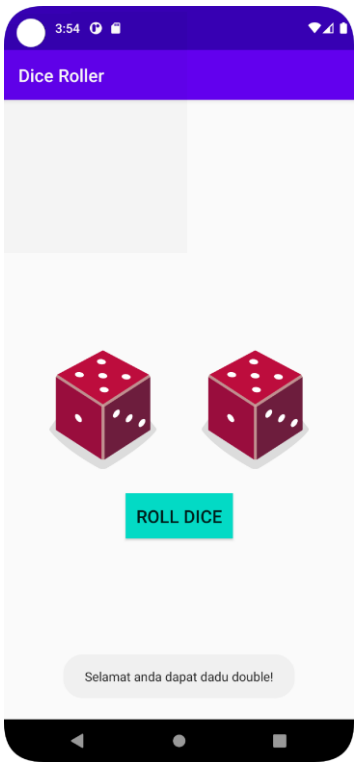
Gambar 1. Soal 1 Modul 1

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



Gambar 2. Soal 1 Modul 1

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



Gambar 3. Soal 1 Modul 1

4. Upload aplikasi yang telah Anda buat ke dalam 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 di download pada link berikut: https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export= download

A. Source Code

```
1  package com.example.myapplication
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.activity.ComponentActivity
6  import androidx.activity.compose.setContent
```

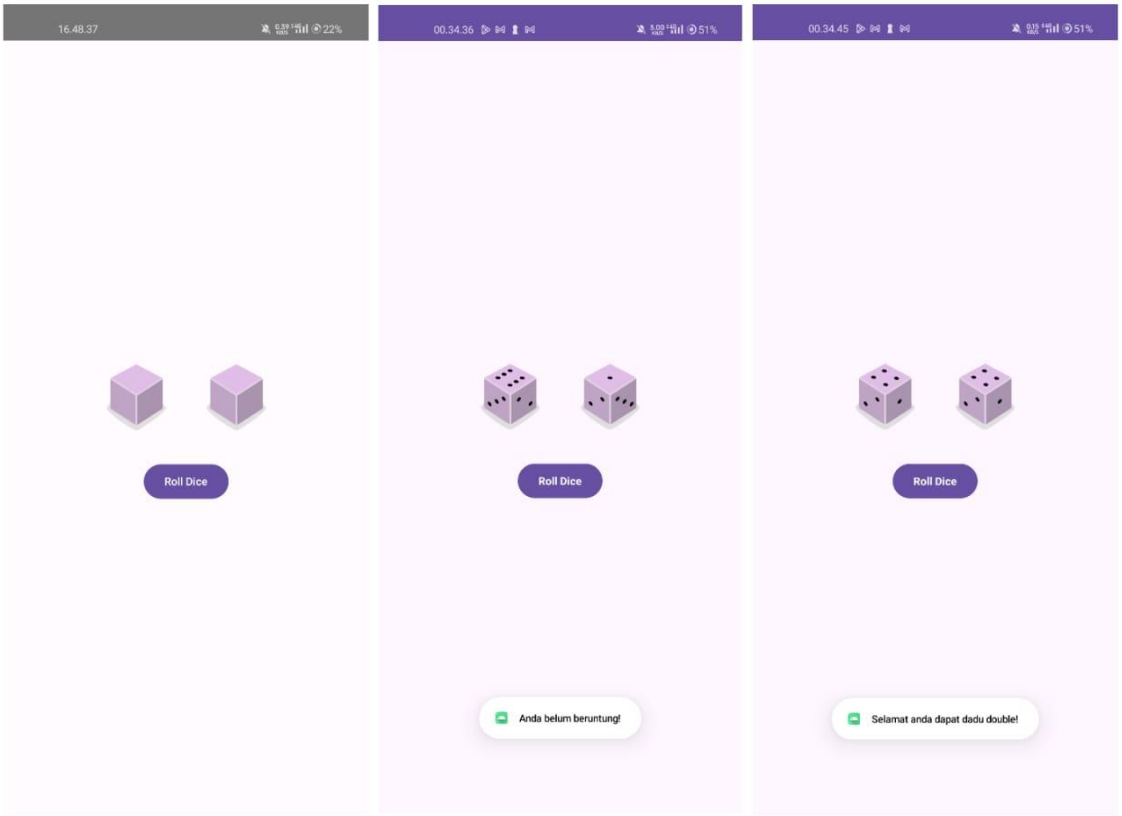
```
7  import androidx.compose.foundation.Image
8  import androidx.compose.foundation.layout.*
9  import androidx.compose.material3.*
10 import androidx.compose.runtime.*
11 import androidx.compose.ui.Alignment
12 import androidx.compose.ui.Modifier
13 import androidx.compose.ui.platform.LocalContext
14 import androidx.compose.ui.res.painterResource
15 import androidx.compose.ui.unit.dp
16 import kotlin.random.Random
17
18 class MainActivity : ComponentActivity() {
19     override fun onCreate(savedInstanceState: Bundle?)
20 {
21         super.onCreate(savedInstanceState)
22         setContent {
23             Surface(
24                 modifier = Modifier.fillMaxSize(),
25                 color                                    =
26 MaterialTheme.colorScheme.background
27             ) {
28                 DiceRoller()
29             }
30         }
31     }
32 }
33
34 @Composable
35 fun DiceRoller() {
36     var dice1 by remember { mutableStateOf(0) }
37     var dice2 by remember { mutableStateOf(0) }
```

```
38      val context = LocalContext.current
39
40      Column(
41          modifier = Modifier
42              .fillMaxSize()
43              .padding(16.dp),
44          verticalArrangement = Arrangement.Center,
45          horizontalAlignment                                =
46  Alignment.CenterHorizontally
47      ) {
48          Row(
49              horizontalArrangement                          =
50  Arrangement.spacedBy(16.dp),
51              verticalAlignment                              =
52  Alignment.CenterVertically
53          ) {
54              DiceImage(dice1)
55              DiceImage(dice2)
56          }
57
58          Spacer(modifier = Modifier.height(24.dp))
59
60          Button(onClick = {
61              dice1 = Random.nextInt(1, 7)
62              dice2 = Random.nextInt(1, 7)
63
64              if (dice1 == dice2) {
65                  Toast.makeText(context, "Selamat anda
66  dapat dadu double!", Toast.LENGTH_SHORT).show()
67              } else {
68
```

```
69                  Toast.makeText(context,   "Anda   belum
70 beruntung!", Toast.LENGTH_SHORT).show()
71              }
72          }) {
73              Text("Roll Dice")
74          }
75      }
76 }
77
78 @Composable
79 fun DiceImage(diceValue: Int) {
80      val imageResource = when (diceValue) {
81          1 -> R.drawable.dice_1
82          2 -> R.drawable.dice_2
83          3 -> R.drawable.dice_3
84          4 -> R.drawable.dice_4
85          5 -> R.drawable.dice_5
86          6 -> R.drawable.dice_6
87          else -> R.drawable.dice_0
88      }
89
90      Image(
91          painter = painterResource(id = imageResource),
92          contentDescription = "Gambar Dadu",
93          modifier = Modifier.size(100.dp)
94      )
95 }
```

Tabel 1. Source Code Soal 1 Modul 1

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

Pada line 1, package com.example.myapplication menyatakan bahwa file ini berada di dalam package com.example.myapplication.

Pada line 3–9, Bagian import digunakan untuk:

- Mengakses elemen penting Compose seperti `Button, Image, Column,` dll.
- `Toast` untuk menampilkan pesan pop-up.
- `LocalContext` untuk mendapatkan Context saat ini.
- `Random` untuk menghasilkan angka acak dadu.

Pada line 11, `class MainActivity : ComponentActivity()` adalah activity utama yang mewarisi `ComponentActivity`, digunakan saat membuat UI dengan Jetpack Compose.

Pada line 12–18, `onCreate()` digunakan untuk membuat tampilan saat activity pertama kali dijalankan:

- `setContent { ... }` menandakan awal dari Compose UI.
- `Surface(...)` digunakan sebagai kontainer latar belakang.
- `DiceRoller()` adalah composable function yang berisi tampilan dan logika utama.

Pada line 21–23:

- `var dice1 by remember { mutableStateOf(0) }` dan `dice2` menyimpan nilai dadu, dan akan berubah secara dinamis jika ditekan tombol.
- `val context = LocalContext.current` digunakan untuk menampilkan Toast pada konteks sekarang.

Pada line 25–32, `Column` menyusun elemen UI secara vertikal dan berada di tengah layar.

- `modifier.fillMaxSize()` agar elemen mengisi seluruh layar.
- `padding(16.dp)` memberi jarak dari tepi layar.
- `verticalArrangement = Arrangement.Center` memusatkan secara vertikal.
- `horizontalAlignment = Alignment.CenterHorizontally` memusatkan secara horizontal.

Pada line 33–37, `Row` digunakan untuk menempatkan dua gambar dadu secara horizontal.

- `Arrangement.spacedBy(16.dp)` memberi jarak antar dadu.
- Memanggil fungsi `DiceImage(dice1)` dan `DiceImage(dice2)`.

Pada line 39, `Spacer(modifier = Modifier.height(24.dp))` memberi jarak vertikal antara gambar dadu dan tombol.

Pada line 41–47, `Button` digunakan untuk mengacak dadu:

- `Random.nextInt(1, 7)` menghasilkan angka 1–6 untuk kedua dadu.
- `if (dice1 == dice2)` akan menampilkan Toast jika dadu double.
- Jika tidak sama, akan muncul Toast “Anda belum beruntung!”.

Pada line 51–59:

- `val imageResource = when (diceValue) { ... }` menentukan gambar dadu berdasarkan angka yang dihasilkan (1–6).
- Jika nilai belum valid (sebelum tombol ditekan), akan ditampilkan gambar `dice_0`.

Pada line 61–65, `Image(...)` menampilkan gambar dadu:

- `painterResource(id = imageResource)` memuat gambar dari resource.
- `contentDescription = "Gambar Dadu"` untuk aksesibilitas.
- `modifier = Modifier.size(100.dp)` mengatur ukuran dadu 100x100 dp

MODUL 2 : Android Layout

SOAL 1

Buatlah program yang dapat mencetak kalimat “Hello World in PHP” menggunakan Bahasa pemrograman PHP.

A. Source Code

1. MainActivity.kt

```
1  package com.example.kalkulator
2
3  import android.os.Bundle
4  import android.widget.*
5  import androidx.appcompat.app.AppCompatActivity
6  import java.text.NumberFormat
7  import kotlin.math.ceil
8
9  class MainActivity : AppCompatActivity() {
10     override fun onCreate(savedInstanceState: Bundle?)
11 {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         val                  etCost                  =
16 findViewById<EditText>(R.id.etCost)
17         val                rgTipOptions              =
18 findViewById<RadioGroup>(R.id.rgTipOptions)
19         val               switchRoundUp              =
20 findViewById<Switch>(R.id.switchRoundUp)
```

```
        val                btnCalculate                =
findViewById<Button>(R.id.btnCalculate)
        val                  tvResult                  =
findViewById<TextView>(R.id.tvResult)

        btnCalculate.setOnClickListener {
            val                  cost                  =
etCost.text.toString().toDoubleOrNull()

            if (cost == null || cost == 0.0) {
                tvResult.text = "Tip Amount: $0.00"
                return@setOnClickListener
            }

            val        tipPercent        =        when
(rgTipOptions.checkedRadioButtonId) {
                R.id.rbAmazing -> 0.20
                R.id.rbGood -> 0.18
                else -> 0.15
            }

            var tip = cost * tipPercent
            if (switchRoundUp.isChecked) {
                tip = ceil(tip)
            }

            val             formattedTip             =
NumberFormat.getCurrencyInstance().format(tip)
            tvResult.text = "Tip Amount: $formattedTip"
        }
    }
```

```
52  }
53
```

Tabel 2. Source Code Soal 1 Modul 2

2. activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
3   xmlns:android="http://schemas.android.com/apk/res/andr
4   oid"
5       android:orientation="vertical"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:background="@android:color/white"
9       android:fitsSystemWindows="true">
10
11      <TextView
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:background="#6200EE"
15          android:text="Tip Time"
16          android:textColor="@android:color/white"
17          android:gravity="left"
18          android:textSize="20sp"
19          android:padding="16dp" />
20
21      <ScrollView
22          android:layout_width="match_parent"
23          android:layout_height="match_parent">
24
25          <LinearLayout
26              android:layout_width="match_parent"
27              android:layout_height="wrap_content"
```

```
28              android:orientation="vertical"
29              android:padding="24dp">
30
31              <EditText
32                  android:id="@+id/etCost"
33                  android:layout_width="match_parent"
34                  android:layout_height="wrap_content"
35                  android:hint="Cost of Service"
36                  android:inputType="numberDecimal"
37                  android:backgroundTint="#6200EE"
38                  android:textColor="#000000"
39                  android:textColorHint="#999999"
40                  android:layout_marginTop="16dp" />
41
42              <TextView
43                  android:layout_width="wrap_content"
44                  android:layout_height="wrap_content"
45                  android:text="How was the service?"
46                  android:textColor="#888888"
47                  android:layout_marginTop="16dp" />
48
49              <RadioGroup
50                  android:id="@+id/rgTipOptions"
51                  android:layout_width="match_parent"
52                  android:layout_height="wrap_content">
53
54                  <RadioButton
55                      android:id="@+id/rbAmazing"
56
57  android:layout_width="wrap_content"
58
```

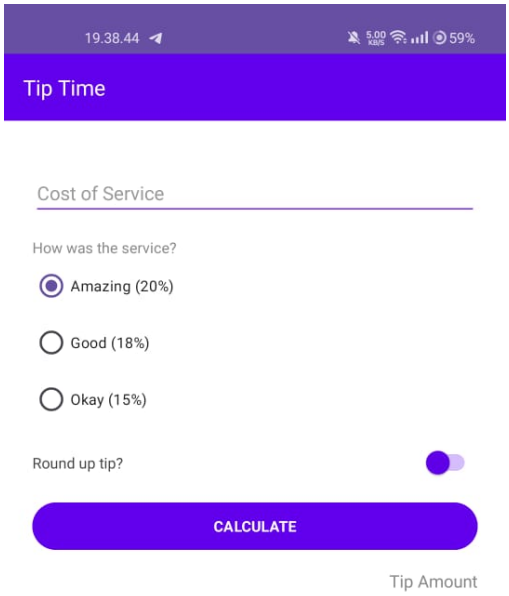
```
59
60  android:layout_height="wrap_content"
61                      android:text="Amazing (20%)"
62                      android:checked="true" />
63
64                  <RadioButton
65                      android:id="@+id/rbGood"
66
67  android:layout_width="wrap_content"
68
69  android:layout_height="wrap_content"
70                      android:text="Good (18%)" />
71
72                  <RadioButton
73                      android:id="@+id/rbOkay"
74
75  android:layout_width="wrap_content"
76
77  android:layout_height="wrap_content"
78                      android:text="Okay (15%)" />
79              </RadioGroup>
80
81              <LinearLayout
82                  android:layout_width="match_parent"
83                  android:layout_height="wrap_content"
84                  android:orientation="horizontal"
85                  android:gravity="center_vertical"
86                  android:layout_marginTop="16dp">
87
88                  <TextView
89                      android:layout_width="0dp"
```

```
90
91  android:layout_height="wrap_content"
92                      android:text="Round up tip?"
93                      android:layout_weight="1"
94                      android:textColor="#444444" />
95
96                  <Switch
97                      android:id="@+id/switchRoundUp"
98
99  android:layout_width="wrap_content"
100
101 android:layout_height="wrap_content"
102                     android:trackTint="#6200EE"
103                     android:thumbTint="#6200EE" />
104             </LinearLayout>
105
106             <Button
107                 android:id="@+id/btnCalculate"
108                 android:layout_width="match_parent"
109                 android:layout_height="wrap_content"
110                 android:text="CALCULATE"
111                 android:layout_marginTop="16dp"
112                 android:backgroundTint="#6200EE"
113
114 android:textColor="@android:color/white" />
115
116             <TextView
117                 android:id="@+id/tvResult"
118                 android:layout_width="match_parent"
119                 android:layout_height="wrap_content"
120                 android:text="Tip Amount"
```

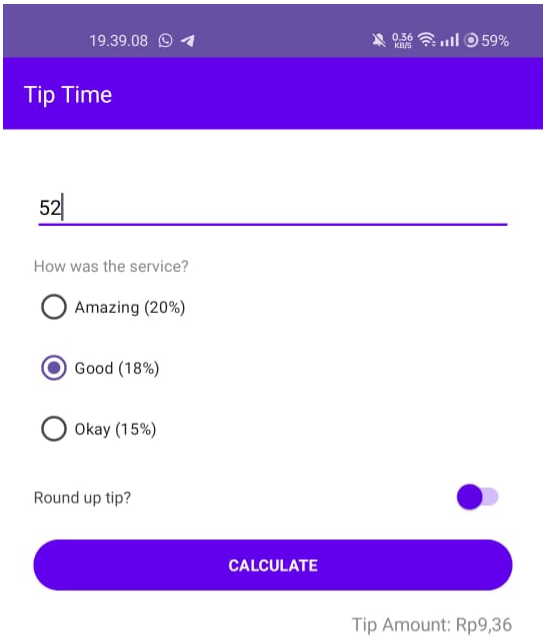
```
121                 android:layout_marginTop="12dp"
122                 android:gravity="end"
123                 android:textColor="#888888"
124                 android:textSize="16sp" />
125         </LinearLayout>
126     </ScrollView>
    </LinearLayout>
```

Tabel 3. Source Code Soal 1 Modul 2

B. Output Program



Gambar 5. Screenshot Hasil Jawaban Soal 1 Modul 2



Gambar 6. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

1. MainActivity.kt

Pada baris 1, dideklarasikan nama package file Kotlin, yaitu `com.example.kalkulator`. Ini menunjukkan bahwa file `MainActivity.kt` berada dalam package tersebut. Pada baris 3, diimpor class

`Bundle` yang digunakan untuk menyimpan dan mengirim data antar activity, khususnya saat proses lifecycle seperti `onCreate`. Pada baris 4, diimpor class `Button` bersama komponen antarmuka pengguna lainnya (`EditText, RadioGroup, Switch, dan TextView`) melalui `android.widget.*`, agar komponen-komponen tersebut dapat digunakan dalam kode Kotlin. Pada baris 5, `AppCompatActivity` diimpor agar class `MainActivity` dapat menggunakan fitur activity modern dan kompatibel dengan berbagai versi Android. Pada baris 6, diimpor `NumberFormat` dari package `java.text` yang berguna untuk mengubah angka menjadi format mata uang.

Pada baris 7, fungsi `ceil()` dari Kotlin diimpor untuk melakukan pembulatan ke atas dalam perhitungan tip. Pada baris 9–31, class `MainActivity` didefinisikan sebagai turunan dari `AppCompatActivity`. Di dalamnya terdapat fungsi `onCreate()` yang dieksekusi saat activity pertama kali dijalankan. Pada baris 11, digunakan `setContentView(R.layout.activity_main)` untuk menampilkan layout XML `activity_main.xml` sebagai antarmuka aplikasi. Selanjutnya, pada baris 13–17, dilakukan inisialisasi komponen tampilan seperti `EditText, RadioGroup, Switch, Button, dan TextView` menggunakan `findViewById`.

Pada baris 19, terdapat event listener `setOnClickListener` yang dipasang pada tombol `CALCULATE`. Ketika tombol ditekan, pada baris 20–23, sistem akan membaca nilai input biaya layanan dari `EditText` dan mengubahnya menjadi tipe `Double`. Jika input kosong atau bernilai nol, maka program akan menampilkan teks `"Tip Amount: $0.00"` pada `TextView` dan menghentikan eksekusi kode menggunakan `return`.

Pada baris 25–28, dilakukan pengecekan terhadap radio button yang dipilih oleh pengguna untuk menentukan persentase tip: 20% (`rbAmazing`), 18% (`rbGood`), dan 15% sebagai default. Nilai ini kemudian digunakan untuk menghitung tip pada baris 30. Jika pengguna mengaktifkan `Switch` pembulatan, maka nilai tip akan dibulatkan ke atas pada baris 31 menggunakan fungsi `ceil`. Terakhir, pada baris 33–34, nilai tip yang sudah dihitung diformat menjadi bentuk

mata uang menggunakan `NumberFormat.getCurrencyInstance().format(tip)`, dan hasilnya ditampilkan ke `TextView` sebagai teks `Tip Amount: $[nilai tip]`.

2. activity_main.xml

Pada baris 1, file Kotlin ini dideklarasikan dengan nama package `com.example.kalkulator`, yang menunjukkan lokasi file `MainActivity.kt` dalam struktur proyek. Pada baris 3, diimpor class `Bundle`, yang digunakan untuk menyimpan dan mengirim data antar activity, terutama saat proses lifecycle seperti `onCreate`. Selanjutnya, pada baris 4, beberapa komponen antarmuka pengguna diimpor dari `android.widget.*`, seperti `Button`, `EditText`, `RadioGroup`, `Switch`, dan `TextView`, yang diperlukan untuk interaksi dengan pengguna dalam aplikasi.

Pada baris 5, `AppCompatActivity` diimpor agar `MainActivity` dapat mengakses berbagai fitur modern dari activity dan kompatibel dengan berbagai versi Android. Baris 6 mengimpor `NumberFormat` dari package `java.text`, yang berguna untuk memformat angka menjadi format mata uang. Sementara itu, pada baris 7, fungsi `ceil()` dari Kotlin diimpor untuk melakukan pembulatan ke atas, yang digunakan dalam perhitungan tip.

Pada baris 9 hingga 31, didefinisikan class `MainActivity` yang merupakan turunan dari `AppCompatActivity`. Di dalamnya, terdapat fungsi `onCreate()` yang dieksekusi saat activity pertama kali dijalankan. Pada baris 11, `setContentView(R.layout.activity_main)` digunakan untuk menampilkan layout XML `activity_main.xml`, yang berfungsi sebagai antarmuka aplikasi. Pada baris 13 hingga 17, dilakukan inisialisasi komponen-komponen tampilan seperti `EditText`, `RadioGroup`, `Switch`, `Button`, dan `TextView` menggunakan `findViewById`.

Di baris 19, sebuah event listener `setOnClickListener` dipasang pada tombol `CALCULATE`. Ketika tombol ditekan, pada baris 20 hingga 23, nilai input biaya layanan yang dimasukkan oleh pengguna akan dibaca dari `EditText` dan

diubah menjadi tipe `Double`. Jika input kosong atau bernilai nol, aplikasi akan menampilkan teks `"Tip Amount: $0.00"` di `TextView` dan menghentikan eksekusi kode menggunakan `return`.

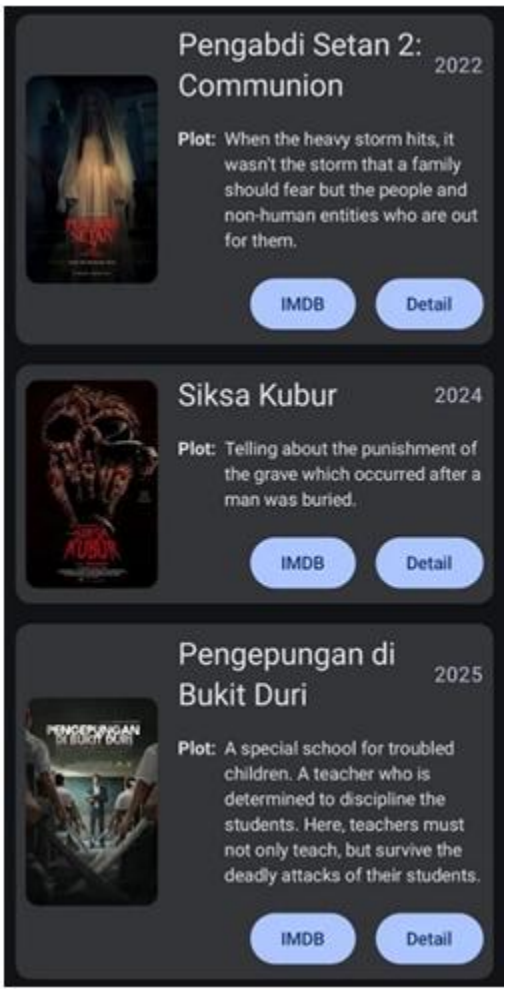
Pada baris 25 hingga 28, dilakukan pengecekan terhadap radio button yang dipilih untuk menentukan persentase tip: 20% untuk pilihan `rbAmazing`, 18% untuk pilihan `rbGood`, dan 15% untuk pilihan default. Nilai tip ini kemudian digunakan dalam perhitungan pada baris 30. Jika pengguna mengaktifkan `Switch` untuk pembulatan, nilai tip akan dibulatkan ke atas pada baris 31 menggunakan fungsi `ceil()`. Akhirnya, pada baris 33 hingga 34, nilai tip yang telah dihitung diformat menjadi format mata uang menggunakan `NumberFormat.getCurrencyInstance().format(tip)` dan ditampilkan ke dalam `TextView` sebagai teks yang berbunyi `Tip Amount: $[nilai tip]`.

MODUL 3 : Build a Scrollable List

SOAL 1

1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
   a. List menggunakan fungsi RecylerView (XML) atau LazyColumn (Compose).
   b. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas.
   c. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah.
   d. Terdapat 2 button dalam list, dengan fungsi sebagai berikut:
      - Button pertama menggunakan intent eksplisit untuk membuka apliaksi atau browser lain.
      - Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
   e. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius.
   f. Saat orientasi perangkat berubah/di rotasi, baik ke potrait maupun landscape, aplikasi responsif dan dpaat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang.
   g. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment).
   h. Aplikasi berbasis XML harus menggunakan ViewBinding.
2. Mengapa RecyclerView masih digunakan, padahal RecyclerView memiliki kode yang panjang dan bersifat boiler-plate, dibandingkan LazyColumn dengan kode yang lebih singkat?

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



Gambar 7. Soal 1 Modul 3

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

Gambar 8. Soal 1 Modul 3

A. Source Code

1. MainActivity

```
1  package com.example.mlacharacters
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.compose.material3.MaterialTheme
7  import androidx.compose.material3.Surface
8  import
9  androidx.navigation.compose.rememberNavController
10 import
11 com.example.mlacharacters.ui.theme.MLACharactersTheme
12
```

```
13 class MainActivity : ComponentActivity() {
14     override fun onCreate(savedInstanceState: Bundle?)
15 {
16         super.onCreate(savedInstanceState)
17         setContent {
18             MLACharactersTheme {
19                 Surface(color                          =
20 MaterialTheme.colorScheme.background) {
21                     val         navController         =
22 rememberNavController()
23                     NavGraph(navController             =
24 navController)
25                 }
26             }
27         }
28     }
29 }
30
```

Tabel 4. Source Code Soal 1 Modul 3

2. Character

```
1 package com.example.mlacharacters
2
3 data class Character(
4     val name: String,
5     val alias: String,
6     val imageRes: Int,
7     val description: String,
8     val wikiUrl: String
9 )
```

Tabel 5. Source Code Soal 1 Modul 3

3. CharacterDetailScreen.kt

```
1  package com.example.mlacharacters
2
3  import androidx.compose.foundation.Image
4  import androidx.compose.foundation.layout.*
5  import androidx.compose.foundation.lazy.LazyColumn
6  import androidx.compose.material.icons.Icons
7  import
8  androidx.compose.material.icons.filled.ArrowBack
9  import androidx.compose.material3.*
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.graphics.Color
13 import androidx.compose.ui.layout.ContentScale
14 import androidx.compose.ui.res.painterResource
15 import androidx.compose.ui.text.font.FontWeight
16 import androidx.compose.ui.unit.dp
17 import androidx.compose.ui.unit.sp
18 import androidx.navigation.NavController
19
20 @OptIn(ExperimentalMaterial3Api::class)
21 @Composable
22 fun    CharacterDetailScreen(character:    Character,
23 navController: NavController) {
24     Scaffold(
25         containerColor = Color(0xFF1C1C1E),
26         topBar = {
27             CenterAlignedTopAppBar(
28                 title = {},
29                 navigationIcon = {
30                     IconButton(onClick       =       {
31 navController.popBackStack() }) {
```

```
                        Icon(
                            imageVector             =
Icons.Default.ArrowBack,
                            contentDescription      =
"Back",
                            tint = Color.White
                        )
                    }
                },
                colors                                  =
TopAppBarDefaults.centerAlignedTopAppBarColors(
                    containerColor = Color.Transparent
                )
            )
        }
    ) { innerPadding ->
        LazyColumn(
            modifier = Modifier
                .padding(innerPadding)
                .fillMaxSize()
        ) {
            item {
                Image(
                    painter  =  painterResource(id  =
character.imageRes),
                    contentDescription              =
character.name,
                    contentScale = ContentScale.Crop,
                    modifier = Modifier
                        .fillMaxWidth()
                        .height(500.dp)
```

```
                )
            }

            item {
                Column(modifier = Modifier.padding(16.dp)) {
                    Row(
                        modifier = Modifier.fillMaxWidth(),
                        horizontalArrangement = Arrangement.SpaceBetween
                    ) {
                        Text(
                            text = character.name,
                            color = Color.White,
                            style = MaterialTheme.typography.titleLarge,
                            fontWeight = FontWeight.Bold
                        )
                        Text(
                            text = "2016",
                            color = Color.LightGray,
                            style = MaterialTheme.typography.bodyMedium
                        )
                    }

                    Spacer(modifier = Modifier.height(8.dp))

```

```
94                      Text(
95                          text = "Deskripsi:",
96                          color = Color.White,
97                          fontWeight = FontWeight.Bold,
98                          style                          =
99  MaterialTheme.typography.bodyMedium
100                     )
101
102                     Spacer(modifier                    =
103 Modifier.height(4.dp))
104
105                     Text(
106                         text = character.description,
107                         color = Color(0xFFCCCCCC),
108                         style                          =
109 MaterialTheme.typography.bodySmall,
110                         lineHeight = 20.sp
111                     )
112                 }
113             }
114         }
115     }
116 }
117
118
```

Tabel 6. Source Code Soal 1 Modul 3

4. CharacterListScreen.kt

```
1   package com.example.mlacharacters
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
```

```
5  import androidx.compose.foundation.lazy.LazyColumn
6  import androidx.compose.foundation.lazy.items
7  import androidx.compose.material3.*
8  import androidx.compose.runtime.Composable
9  import
10 androidx.compose.foundation.shape.RoundedCornerShape
11 import androidx.compose.ui.text.font.FontWeight
12 import androidx.compose.ui.Alignment
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.graphics.Color
15 import androidx.compose.ui.layout.ContentScale
16 import androidx.compose.ui.res.painterResource
17 import androidx.compose.ui.unit.dp
18 import androidx.navigation.NavController
19 import android.content.Intent
20 import android.net.Uri
21 import androidx.compose.ui.platform.LocalContext
22
23 @OptIn(ExperimentalMaterial3Api::class)
24 @Composable
25 fun  CharacterListScreen(characters:  List<Character>,
26 navController: NavController) {
27     LazyColumn(
28
29         modifier = Modifier
30             .fillMaxSize()
31             .padding(8.dp),
32
33         contentPadding = PaddingValues(top = 60.dp),
34         verticalArrangement                          =
35 Arrangement.spacedBy(12.dp)
```

```
36
37      ) {
38          items(characters) { character ->
39              Card(
40                  shape = RoundedCornerShape(12.dp),
41                  colors                                    =
42  CardDefaults.cardColors(containerColor                    =
43  Color(0xFF2B2B2B)),
44                  elevation                                 =
45  CardDefaults.cardElevation(4.dp),
46                  modifier = Modifier.fillMaxWidth()
47              ) {
48                  Row(modifier                              =
49  Modifier.padding(12.dp)) {
50                      Image(
51                          painter = painterResource(id =
52  character.imageRes),
53                          contentDescription                =
54  character.name,
55                          contentScale                      =
56  ContentScale.Crop,
57
58                          modifier = Modifier
59                              .width(90.dp)
60                              .height(140.dp)
61                              .padding(end = 12.dp)
62
63                      )
64
65                      Column(modifier                       =
66  Modifier.weight(1f)) {
```

```
                        Row(
                            modifier                =
Modifier.fillMaxWidth(),
                            horizontalArrangement   =
Arrangement.SpaceBetween
                        ) {
                            Text(
                                text = character.name,
                                style                =
MaterialTheme.typography.titleMedium,
                                color = Color.White,
                                fontWeight           =
FontWeight.Bold
                            )
                            Text(
                                text = "2016",
                                style                =
MaterialTheme.typography.labelMedium,
                                color                =
Color.LightGray
                            )
                        }

                        Spacer(modifier              =
Modifier.height(6.dp))

                        Row(
                            modifier                =
Modifier.fillMaxWidth(),
                            verticalAlignment       =
Alignment.Top
```

```
98                          ) {
99                              Text(
100                                 text = "Deskripsi: ",
101                                 color = Color.White,
102                                 fontWeight          =
103 FontWeight.Bold,
104                                 style               =
105 MaterialTheme.typography.bodyMedium
106                             )
107
108                             Spacer(modifier         =
109 Modifier.width(8.dp))
110
111                             Text(
112                                 text                =
113 character.description,
114                                 color = Color.White,
115                                 style               =
116 MaterialTheme.typography.bodySmall,
117                                 modifier            =
118 Modifier.weight(1f)
119                             )
120                         }
121
123                         Spacer(modifier             =
124 Modifier.height(10.dp))
125
126                         Row(
127                             horizontalArrangement   =
128 Arrangement.spacedBy(8.dp)
129                         ) {
```

```
130                             val        context        =
131 LocalContext.current
132
134                             Button(
135                                 onClick = {
136                                     val    intent    =
137 Intent(Intent.ACTION_VIEW,
138 Uri.parse(character.wikiUrl))
139
140 context.startActivity(intent)
141                                 },
142                                 colors                =
143 ButtonDefaults.buttonColors(containerColor            =
144 Color(0xFF9BB1EB)),
145                                 modifier              =
146 Modifier.weight(1f)
147                             ) {
148                                 Text("Detail    Hero",
149 color = Color.White)
150                             }
151
152                             Button(
153                                 onClick = {
154
155 navController.navigate("detail/${character.name}")
156                                 },
157                                 colors                =
158 ButtonDefaults.buttonColors(containerColor            =
159 Color(0xFF9BB1EB)),
160                                 modifier              =
161 Modifier.weight(1f)
```

```
                                ) {
                                    Text("Deskripsi",
color = Color.White)
                                }
                            }
                        }
                    }
                }
            }
        }
```

Tabel 7. Source Code Soal 1 Modul 3

5. NavGraph.kt

```
package com.example.mlacharacters

import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable

@Composable
fun NavGraph(navController: NavHostController) {
    val characters = listOf(
        Character("Crocell", "Mage, Order: Light
Equipment", R.drawable.crocell, "Role: Mage\nType:
Order - Light Equipment\n\nSaya menyukai hero ini
karena tampilannya yang sangat cantik dan positif dan
voicenya yang ceria, menarik serta lucu. Saya paling
menyukai voice line ketika menang, 'Horay, we won!'.
```

```
Ultimate Crocell juga berguna untuk team fight dan efek ultimatenya terang dan bagus, seperti bintang yang jatuh di langit dengan sukacita.", "https://mla.fandom.com/wiki/Crocell"),
        Character("Astraia Sipra", "Support, Hybrid, Light - Tech: Light Equipment", R.drawable.sipra, "Role: Support\nType: Hybrid (Light - Tech) - Light Equipment\n\nDikenal dengan panggilan Sipra. Hero ini juga termasuk dalam favorit saya karena ultimatenya yang sangat berguna untuk war. Bahkan Sipra B4 bisa mengimbangi war hero yang sudah awakened. Ultimatenya adalah meng-summon hero secara acak.", "https://mla.fandom.com/wiki/Astraia_Sipra"),
        Character("Nana", "Spirit, Martial: Light Equipment", R.drawable.nana, "Role: Spirit\nType: Martial - Light Equipment\n\nNana di Mobile Legends Bang Bang dengan Mobile Legends Adventure sangat berbeda. Jujur ketika saya pertama kali memakai hero ini, skill nya sangat berbeda dengan yang versi moba nya. Skill 2 disini adalah memberi heal pada tim dan ultimatenya membangkitkan hero yang sudah mati, atau me-respawn hero secara langsung ketika ultimatenya siap.", "https://mla.fandom.com/wiki/Nana"),
        Character("Mecha Layla", "Marksman, Hybrid (Light - Tech): Light Equipment", R.drawable.mechalayla, "Role: Marksman\nType: Hybrid (Light - Tech) - Light Equipment\n\nLayla di MLA ada 2 versi, yang pertama adalah Layla dan yang kedua adalah Mecha Layla. Hero menurut saya juga cukup meta karena memiliki kemampuan heal kepada tim, dan pasifnya yang tidak dapat diserang oleh lawan (invisible). Jadi,
```

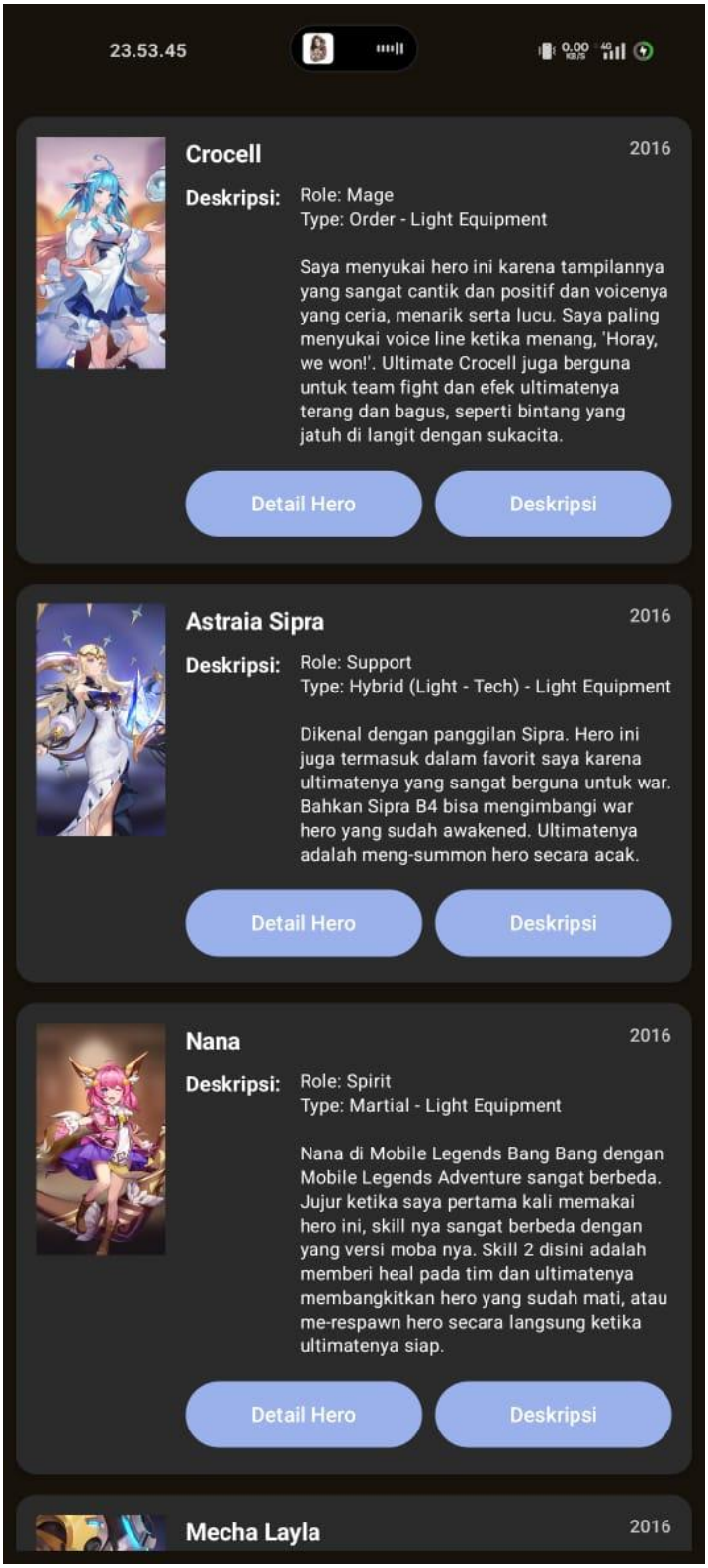
```
Mecha Layla akan memberi damage dan heal secara terus-
menerus tanpa tereteksi oleh musuh. Namun, ketika semua
hero di tim telah lenyap, maka Mecha Layla juga ikut
lenyap", "https://mla.fandom.com/wiki/Mecha_Layla"),
        Character("Natsu", "Fighter, Hybrid (Dark -
Martial): Medium Equipment", R.drawable.natsu, "Role:
Fighter\nType:Hybrid (Dark - Martial) - Medium
Equipment\n\nNatsu, hero kolaborasi dengan anime Fairy
Tail. Natsu dan Lucy dengan status card SSR, dan Erza
dengan status card UR. Saya mendapatkan card Natsu
secara gratis melalui event, hanya cukup menjalankan
quest saya bisa mendapatkan Natsu dengan B5 merah (belum
awakened). Yang saya sukai dari hero ini adalah ketika
posisi back line diserang, Natsu dengan cepat
menghampiri dan menyerang serta ultimatenya yang
mendorong semua musuh ke kanan, dimana ini sangat
menguntungkan tim terutama jika dikombo dengan Alice
atau hero tipe AoE. Entah ini kebetulan atau bukan,
tapi saya juga seringkali melihat ketika Natsu
mengeluarkan ultimate pada HP di bawah 15%, HP dia akan
terisi secara penuh kembali (kekebalan tubuh)",
"https://mla.fandom.com/wiki/Natsu"),
        Character("Naiad Rafaela", "Hybrid (Light -
Elemental): Light Equipment", R.drawable.rafaela,
"Role:Support\nType: Hybrid (Light - Elemental) - Light
Equipment\n\nVersi lain dari hero Rafaela. Saya tidak
memperhatikan skill-skillnya apakah berguna untuk war
atau tidak. Namun, ketika saya pertama kali melihat
hero SSR ini, saya langsung menyukainya karena
desainnya yang begitu cantik, anggun, elegan, dan
```

```
79  sangat nature (menyatu dengan alam). ",
80  "https://mla.fandom.com/wiki/Naiad_Rafaela"),
81          Character("Shah Torre", "Mage, Chaos: Light
82  Equipment", R.drawable.torre, "Role: Mage\nType: Chaos
83  - Light Equipment\n\nShah Torre, hero yang sangat
84  menyebalkan jika menjadi musuh. Ultimate hero ini
85  menyerap bar energi, dimana ketika bar energi sudah
86  penuh, maka hero akan mengeluarkan ultimate mereka.
87  Namun, dengan ultimate Torre ini maka ultimate hero
88  lawan akan tertunda karena ia menyerap bar energi pada
89  semua                                          hero.",
90  "https://mla.fandom.com/wiki/Shah_Torre"),
91          Character("Forseti", "Marksman, Order: Light
92  Equipment", R.drawable.forseti, "Role: Marksman\nType:
93  Order - Light Equipment\n\nSaya baru tahu ada hero ini
94  ketika Tower of Babel khusus hero Order dibuka. Awalnya
95  saya tidak menaruh harapan yang tinggi kepada hero ini,
96  namun ketika melihat ia MVP di hampir semua match, saya
97  memperhatikan skill-skillnya. Yang menarik dari hero
98  ini adalah ultimatenya, ketika Forseti mengeluarkan
99  ultimate maka ia akan mati. Jadi, hero ini kurang cocok
100 jika tidak ada hero lain yang bisa me-respawn hero tim,
101 seperti     Nana     atau     Singularity     Lunox.",
102 "https://mla.fandom.com/wiki/Forseti"),
103
104     )
105     val charMap = characters.associateBy { it.name }
106
107     NavHost(navController, startDestination = "list")
108 {
109         composable("list") {
```

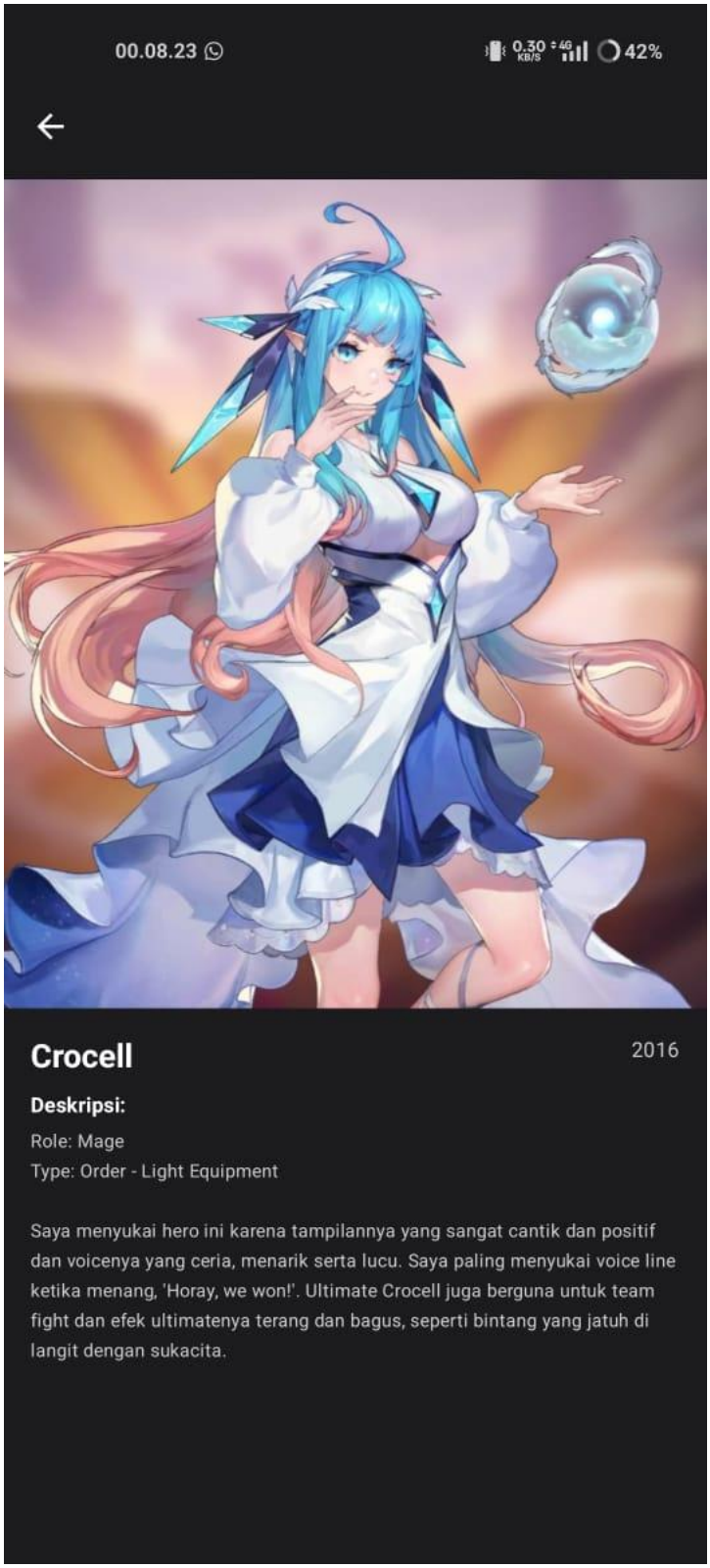
```
110             CharacterListScreen(characters =
111 characters, navController = navController)
112         }
113         composable("detail/{name}") { backStackEntry -
114 >
115             val name =
116 backStackEntry.arguments?.getString("name")
117             val character = charMap[name]
118             character?.let {
119                 CharacterDetailScreen(character = it,
120 navController = navController)
121             }
123         }
124     }
125 }
```

Tabel 8. Source Code Soal 1 Modul 3

B. Output Program



Gambar 9. Screenshot Hasil Jawaban Soal 1 Modul 3



Gambar 10. Screenshot Hasil Jawaban Soal 1 Modul 3

C. Pembahasan

1. MainActivity

Kode `MainActivity` adalah titik awal aplikasi yang menggunakan Jetpack Compose. Di dalam `onCreate`, UI dibangun dengan `setContent`, menerapkan tema `MLACharactersTheme`, dan membungkusnya dalam `Surface` berwarna latar dari tema. Sebuah `NavController` dibuat untuk mengelola navigasi antar layar, lalu `NavGraph(navController)` dipanggil untuk mengatur rute-rute tampilan aplikasi seperti daftar dan detail karakter.

2. Character

Kode tersebut mendefinisikan sebuah `data class` bernama `Character` yang digunakan untuk merepresentasikan data karakter dalam aplikasi Android. Kelas ini memiliki lima properti, yaitu `name` untuk menyimpan nama asli karakter, `alias` untuk nama lain atau julukan, `imageRes` sebagai ID dari resource gambar karakter, `description` untuk menyimpan deskripsi karakter, dan `wikiUrl` sebagai tautan ke halaman wiki karakter tersebut.

3. CharacterDetailScreen.kt

Kode di atas merupakan implementasi tampilan detail karakter dalam aplikasi Android Jetpack Compose. Fungsi `CharacterDetailScreen` menerima data karakter dan `NavController` untuk navigasi. Komponen `Scaffold` digunakan untuk menyusun struktur UI, dengan `TopAppBar` yang memiliki tombol kembali. Isi layar dibungkus dalam `LazyColumn`, yang memungkinkan seluruh konten digulir. Gambar karakter ditampilkan di bagian atas, diikuti oleh informasi seperti nama, tahun, label deskripsi, dan isi deskripsi. Desain ini membuat tampilan detail tetap rapi dan mudah digulir meskipun konten panjang.

4. CharaterListScreen.kt

Kode di atas menampilkan daftar karakter dalam bentuk kartu menggunakan Jetpack Compose. Fungsi `CharacterListScreen` menerima list data `Character` dan `NavController` untuk navigasi. Daftar ini dibungkus dengan

`LazyColumn` agar bisa digulir secara efisien. Setiap item ditampilkan dalam `Card` dengan gambar karakter di sebelah kiri dan informasi karakter di sebelah kanan. Informasi ini mencakup nama, tahun, dan deskripsi. Dua tombol disediakan: satu untuk membuka tautan Wikipedia karakter menggunakan `Intent`, dan satu lagi untuk navigasi ke layar detail karakter. Desain UI dibuat rapi dan responsif dengan pengaturan padding, warna, dan tata letak.

5. NavGraph.kt

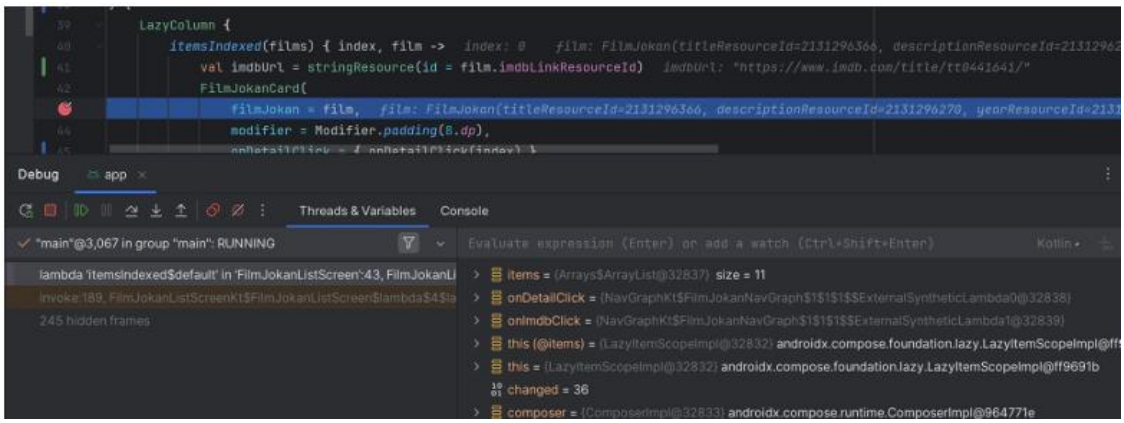
Kode `NavGraph` di atas adalah implementasi navigasi menggunakan Jetpack Compose Navigation. Fungsi ini mendefinisikan dua layar utama dalam aplikasi: `CharacterListScreen` untuk menampilkan daftar karakter dan `CharacterDetailScreen` untuk menampilkan detail karakter berdasarkan nama. Data karakter disusun dalam list `characters`, lalu dipetakan menjadi `charMap` agar bisa diakses dengan cepat berdasarkan nama saat navigasi ke detail. Navigasi dimulai dari route `"list"`, dan ketika user memilih karakter, aplikasi berpindah ke route `"detail/{name}"` menggunakan nama sebagai parameter. Ini memungkinkan perpindahan data antar layar secara dinamis berdasarkan pilihan pengguna.

MODUL 4 : ViewModel dan Debugging

SOAL 1

1. Lanjutkan aplikasi Android berbasis XML atau 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 pembuatam ViewModel.
   c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment.
   d. Gunakan logging untuk event berikut:
      - Log saat data item masuk ke dalam list.
      - Log saat tombol Detail dan tombol Explicit ditekan.
      - Log data dari lisy 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.
2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya.

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



Gambar 11. Soal 1 Modul 4

A. Source Code

1. MainActivity.kt

```
1  package com.example.mlacharacters
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.compose.material3.MaterialTheme
7  import androidx.compose.material3.Surface
8  import androidx.lifecycle.viewmodel.compose.viewModel
9  import
10 androidx.navigation.compose.rememberNavController
11 import com.example.mlacharacters.ui.CharacterViewModel
12 import
13 com.example.mlacharacters.ui.CharacterViewModelFactory
14 import
15 com.example.mlacharacters.ui.theme.MLACharactersTheme
16
17 class MainActivity : ComponentActivity() {
18     override fun onCreate(savedInstanceState: Bundle?)
19 {
```

```
20          super.onCreate(savedInstanceState)
21          setContent {
22              MLACharactersTheme {
23                  Surface(color                          =
24  MaterialTheme.colorScheme.background) {
25                      val         navController         =
26  rememberNavController()
27
28                      val  viewModel:  CharacterViewModel
29  = viewModel(
30                          factory                        =
31  CharacterViewModelFactory()
32                      )
33
34                      NavGraph(navController             =
35  navController, viewModel = viewModel)
36                  }
37              }
38          }
39      }
40  }
```

Tabel 9. Source Code Soal 1 Modul 4

2. Character

```
1  package com.example.mlacharacters
2
3  data class Character(
4      val name: String,
5      val alias: String,
6      val imageRes: Int,
7      val description: String,
8      val wikiUrl: String
```

```
9  )
```

Tabel 10. Source Code Soal 1 Modul 4

3. MLACharacterApplication.kt

```
1  package com.example.mlacharacters
2
3  import android.app.Application
4
5  class MLACharactersApplication : Application() {
6      override fun onCreate() {
7          super.onCreate()
8      }
9  }
```

Tabel 11. Source Code Soal 1 Modul 4

4. CharacterListScreen.kt

```
1  package com.example.mlacharacters
2
3  import android.content.Intent
4  import android.net.Uri
5  import androidx.compose.foundation.Image
6  import androidx.compose.foundation.layout.*
7  import androidx.compose.foundation.lazy.LazyColumn
8  import androidx.compose.foundation.lazy.items
9  import
10 androidx.compose.foundation.shape.RoundedCornerShape
11 import androidx.compose.material3.*
12 import androidx.compose.runtime.Composable
13 import androidx.compose.runtime.collectAsState
14 import androidx.compose.runtime.getValue
15 import androidx.compose.ui.Alignment
16 import androidx.compose.ui.Modifier
```

```
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import com.example.mlacharacters.ui.CharacterViewModel

@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun CharacterListScreen(
    navController: NavController,
    viewModel: CharacterViewModel
) {
    val                  characters                  by
viewModel.characterList.collectAsState()
    val context = LocalContext.current

    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(8.dp),
        contentPadding = PaddingValues(top = 60.dp),
        verticalArrangement                              =
Arrangement.spacedBy(12.dp)
    ) {
        items(characters) { character ->
            Card(
                shape = RoundedCornerShape(12.dp),

```

```
                colors                                  =
CardDefaults.cardColors(containerColor                  =
Color(0xFF2B2B2B)),
                elevation                               =
CardDefaults.cardElevation(4.dp),
                modifier = Modifier.fillMaxWidth()
            ) {
                Row(modifier                            =
Modifier.padding(12.dp)) {
                    Image(
                        painter = painterResource(id =
character.imageRes),
                        contentDescription              =
character.name,
                        contentScale                    =
ContentScale.Crop,
                        modifier = Modifier
                            .width(90.dp)
                            .height(140.dp)
                            .padding(end = 12.dp)
                    )

                    Column(modifier                     =
Modifier.weight(1f)) {
                        Row(
                            modifier                    =
Modifier.fillMaxWidth(),
                            horizontalArrangement       =
Arrangement.SpaceBetween
                        ) {
                            Text(
```

```
                                text = character.name,
                                style                =
MaterialTheme.typography.titleMedium,
                                color = Color.White,
                                fontWeight           =
FontWeight.Bold
                            )
                            Text(
                                text = "2016", // Masih
dummy, bisa pakai data jika tersedia
                                style                =
MaterialTheme.typography.labelMedium,
                                color                =
Color.LightGray
                            )
                        }

                        Spacer(modifier              =
Modifier.height(6.dp))

                        Row(
                            modifier                 =
Modifier.fillMaxWidth(),
                            verticalAlignment        =
Alignment.Top
                        ) {
                            Text(
                                text = "Deskripsi: ",
                                color = Color.White,
                                fontWeight           =
FontWeight.Bold,
```

```
                                style                =
MaterialTheme.typography.bodyMedium
                            )

                            Spacer(modifier          =
Modifier.width(8.dp))

                            Text(
                                text                 =
character.description,
                                color = Color.White,
                                style                =
MaterialTheme.typography.bodySmall,
                                modifier             =
Modifier.weight(1f)
                            )
                        }

                        Spacer(modifier              =
Modifier.height(10.dp))

                        Row(horizontalArrangement    =
Arrangement.spacedBy(8.dp)) {
                            Button(
                                onClick = {

viewModel.logItemClick(character, "Detail Hero")
                                    val    intent    =
Intent(Intent.ACTION_VIEW,
Uri.parse(character.wikiUrl))

```

```
141
142 context.startActivity(intent)
143                                 },
144                                 colors                =
145 ButtonDefaults.buttonColors(containerColor           =
146 Color(0xFF9BB1EB)),
147                                 modifier              =
148 Modifier.weight(1f)
149                             ) {
150                                 Text("Detail    Hero",
151 color = Color.White)
152                             }
153
154                             Button(
155                                 onClick = {
156
157 viewModel.logItemClick(character, "Deskripsi")
158
159 navController.navigate("detail/${character.name}")
160                                 },
161                                 colors                =
162 ButtonDefaults.buttonColors(containerColor           =
163 Color(0xFF9BB1EB)),
164                                 modifier              =
165 Modifier.weight(1f)
166                             ) {
167                                 Text("Deskripsi",
168 color = Color.White)
169                             }
170                         }
171                     }
```

```
                    }
                }
            }
        }
}

```

Tabel 12. Source Code Soal 1 Modul 4

5. NavGraph.kt

```
package com.example.mlacharacters

import androidx.compose.runtime.Composable
import androidx.navigation.NavHostController
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import com.example.mlacharacters.ui.CharacterViewModel
import
com.example.mlacharacters.ui.theme.CharacterDetailScr
een

@Composable
fun NavGraph(navController: NavHostController,
viewModel: CharacterViewModel) {
    val characters = listOf(
        Character(
            "Crocell", "Mage, Order: Light Equipment",
R.drawable.crocell,
            "Role: Mage\nType: Order - Light
Equipment\n\nSaya menyukai hero ini karena tampilannya
yang sangat cantik dan positif dan voicenya yang ceria,
menarik serta lucu. Saya paling menyukai voice line
ketika menang, 'Horay, we won!'. Ultimate Crocell juga
```

```
berguna untuk team fight dan efek ultimatenya terang
dan bagus, seperti bintang yang jatuh di langit dengan
sukacita.",
            "https://mla.fandom.com/wiki/Crocell"
        ),
        Character(
            "Astraia Sipra", "Support, Hybrid, Light -
Tech: Light Equipment", R.drawable.sipra,
            "Role: Support\nType: Hybrid (Light - Tech)
- Light Equipment\n\nDikenal dengan panggilan Sipra.
Hero ini juga termasuk dalam favorit saya karena
ultimatenya yang sangat berguna untuk war. Bahkan Sipra
B4 bisa mengimbangi war hero yang sudah awakened.
Ultimatenya adalah meng-summon hero secara acak.",

"https://mla.fandom.com/wiki/Astraia_Sipra"
        ),
        Character(
            "Nana",    "Spirit,    Martial:    Light
Equipment", R.drawable.nana,
            "Role:  Spirit\nType:  Martial  -  Light
Equipment\n\nNana di Mobile Legends Bang Bang dengan
Mobile Legends Adventure sangat berbeda. Jujur ketika
saya pertama kali memakai hero ini, skill nya sangat
berbeda dengan yang versi moba nya. Skill 2 disini
adalah memberi heal pada tim dan ultinya membangkitkan
hero yang sudah mati.",
            "https://mla.fandom.com/wiki/Nana"
        ),
        Character(

```

```
            "Mecha Layla", "Marksman, Hybrid (Light - Tech): Light Equipment", R.drawable.mechalayla,
            "Role: Marksman\nType: Hybrid (Light - Tech) - Light Equipment\n\nLayla di MLA ada 2 versi. Mecha Layla memberi heal dan damage secara terus-menerus. Tapi jika semua hero tim mati, Mecha Layla juga ikut lenyap.",
            "https://mla.fandom.com/wiki/Mecha_Layla"
        ),
        Character(
            "Natsu", "Fighter, Hybrid (Dark - Martial): Medium Equipment", R.drawable.natsu,
            "Role: Fighter\nType:Hybrid (Dark - Martial) - Medium Equipment\n\nNatsu hero kolaborasi Fairy Tail. Punya skill dorong musuh ke kanan. Menurut saya sangat berguna saat team fight, apalagi jika dikombo dengan AoE seperti Alice.",
            "https://mla.fandom.com/wiki/Natsu"
        ),
        Character(
            "Naiad Rafaela", "Hybrid (Light - Elemental): Light Equipment", R.drawable.rafaela,
            "Role:Support\nType: Hybrid (Light - Elemental) - Light Equipment\n\nDesain sangat anggun dan menyatu dengan alam. Hero ini saya sukai karena estetikanya meskipun belum terlalu dipakai untuk war.",

"https://mla.fandom.com/wiki/Naiad_Rafaela"
        ),
        Character(

```

```
            "Shah Torre", "Mage, Chaos: Light Equipment", R.drawable.torre,
            "Role: Mage\nType: Chaos - Light Equipment\n\nShah Torre menyerap bar energi musuh, sehingga bisa menunda ultimate lawan. Sangat menyebalkan jika menjadi musuh.",
            "https://mla.fandom.com/wiki/Shah_Torre"
        ),
        Character(
            "Forseti", "Marksman, Order: Light Equipment", R.drawable.forseti,
            "Role: Marksman\nType: Order - Light Equipment\n\nUltimatenya membuat Forseti langsung mati. Butuh support respawn seperti Nana atau Lunox. Saya baru sadar potensi hero ini setelah lihat performanya di Tower of Babel.",
            "https://mla.fandom.com/wiki/Forseti"
        ),
    )

    viewModel.setCharacterList(characters)

    val charMap = characters.associateBy { it.name }

    NavHost(navController = navController, startDestination = "list") {
        composable("list") {
            CharacterListScreen(
                navController = navController,
                viewModel = viewModel
            )
```

```
        }
        composable("detail/{name}") { backStackEntry ->
            val name = backStackEntry.arguments?.getString("name")
            val character = charMap[name]
            character?.let {
                viewModel.logNavigateDetail(it)
                CharacterDetailScreen(character = it, navController = navController)
            }
        }
    }
}
```

Tabel 13. Source Code Soal 1 Modul 4

6. CharacterDetailScreen.kt/ui.theme

```
1   package com.example.mlacharacters.ui.theme
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.lazy.LazyColumn
6   import androidx.compose.material.icons.Icons
7   import
8   androidx.compose.material.icons.filled.ArrowBack
9   import androidx.compose.material3.*
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.graphics.Color
13  import androidx.compose.ui.layout.ContentScale
14  import androidx.compose.ui.res.painterResource
15  import androidx.compose.ui.text.font.FontWeight
16  import androidx.compose.ui.unit.dp
17  import androidx.compose.ui.unit.sp
18  import androidx.navigation.NavController
19  import com.example.mlacharacters.Character
20
21  @OptIn(ExperimentalMaterial3Api::class)
22  @Composable
23  fun    CharacterDetailScreen(character:    Character,
24  navController: NavController) {
25      Scaffold(
26          containerColor = Color(0xFF1C1C1E),
27          topBar = {
28              CenterAlignedTopAppBar(
29                  title = {},
30                  navigationIcon = {
31
```

```
                    IconButton(onClick = {
navController.popBackStack() }) {
                        Icon(
                            imageVector =
Icons.Default.ArrowBack,
                            contentDescription =
"Back",
                            tint = Color.White
                        )
                    }
                },
                colors =
TopAppBarDefaults.centerAlignedTopAppBarColors(
                    containerColor = Color.Transparent
                )
            )
        }
    ) { innerPadding ->
        LazyColumn(
            modifier = Modifier
                .padding(innerPadding)
                .fillMaxSize()
        ) {
            item {
                Image(
                    painter = painterResource(id =
character.imageRes),
                    contentDescription =
character.name,
                    contentScale = ContentScale.Crop,
                    modifier = Modifier
```

```
                        .fillMaxWidth()
                        .height(500.dp)
                )
            }

            item {
                Column(modifier =
Modifier.padding(16.dp)) {
                    Row(
                        modifier =
Modifier.fillMaxWidth(),
                        horizontalArrangement =
Arrangement.SpaceBetween
                    ) {
                        Text(
                            text = character.name,
                            color = Color.White,
                            style =
MaterialTheme.typography.titleLarge,
                            fontWeight =
FontWeight.Bold
                        )
                        Text(
                            text = "2016",
                            color = Color.LightGray,
                            style =
MaterialTheme.typography.bodyMedium
                        )
                    }

```

```
94                      Spacer(modifier                     =
95  Modifier.height(8.dp))
96
97                      Text(
98                          text = "Deskripsi:",
99                          color = Color.White,
100                         fontWeight = FontWeight.Bold,
101                         style                           =
102 MaterialTheme.typography.bodyMedium
103                     )
104
105                     Spacer(modifier                     =
106 Modifier.height(4.dp))
107
108                     Text(
109                         text = character.description,
110                         color = Color(0xFFCCCCCC),
111                         style                           =
112 MaterialTheme.typography.bodySmall,
113                         lineHeight = 20.sp
114                     )
115                 }
116             }
117         }
118     }
119 }
```

Tabel 14. Source Code Soal 1 Modul 4

7. CharacterViewModel.kt/ui.theme

```
1  package com.example.mlacharacters.ui
2
3  import android.util.Log
4  import androidx.lifecycle.ViewModel
5  import androidx.lifecycle.viewModelScope
6  import com.example.mlacharacters.Character
7  import kotlinx.coroutines.flow.MutableStateFlow
8  import kotlinx.coroutines.flow.StateFlow
9  import kotlinx.coroutines.launch
10
11 class CharacterViewModel : ViewModel() {
12
13     private        val        _characterList        =
14 MutableStateFlow<List<Character>>(emptyList())
15     val  characterList:  StateFlow<List<Character>>  =
16 _characterList
17
18     fun  setCharacterList(characters:  List<Character>)
19 {
20         viewModelScope.launch {
21             Log.d("CharacterViewModel",   "Data   item
22 masuk ke dalam list")
23             _characterList.value = characters
24         }
25     }
26
27     fun  logItemClick(character:  Character,  action:
28 String) {
29         Log.d("CharacterViewModel",   "Tombol   $action
30 ditekan untuk ${character.name}")
31     }
```

```
32
33      fun logNavigateDetail(character: Character) {
34          Log.d("CharacterViewModel",    "Navigasi    ke
35  detail karakter: ${character.name}")
36      }
37  }
```

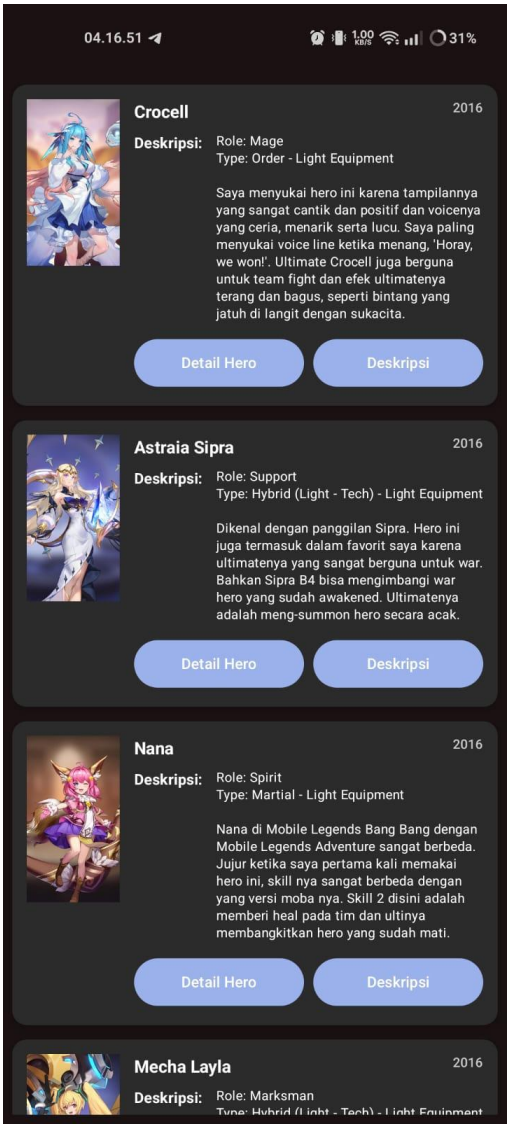
Tabel 15. Source Code Soal 1 Modul 4

8. CharacterViewModelFactory/ui.theme

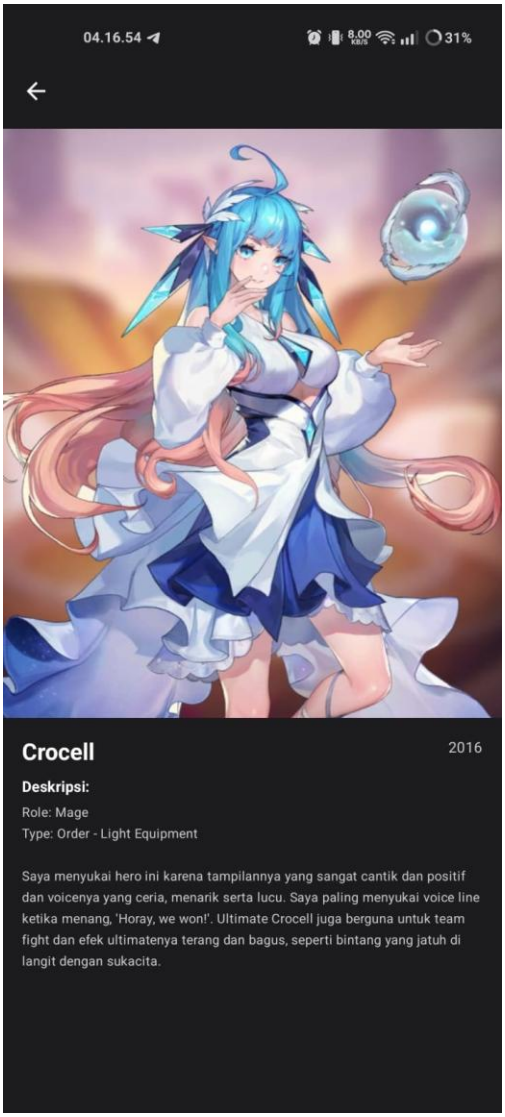
```
1   package com.example.mlacharacters.ui
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class            CharacterViewModelFactory            :
7   ViewModelProvider.Factory {
8       override  fun  <T  :  ViewModel>  create(modelClass:
9   Class<T>): T {
10          if
11  (modelClass.isAssignableFrom(CharacterViewModel::class
12  .java)) {
13              return CharacterViewModel() as T
14          }
15          throw         IllegalArgumentException("Unknown
16  ViewModel class")
17      }
18  }
```

Tabel 16. Source Code Soal 1 Modul 4

B. Output Program



Gambar 12. Screenshot Hasil Jawaban Soal 1 Modul 4



Gambar 13. Screenshot Hasil Jawaban Soal 1 Modul 4

C. Pembahasan

1. Melanjutkan aplikasi Android berbasis Jetpack Compose tanpa mengubah fitur-fitur sebelumnya.
   a. Membuat sebuah ViewModel untuk menyimpan dan mengelola data dari list item.

      ViewModel digunakan untuk menyimpan dan mengelola data UI secara efisien dan aman terhadap perubahan siklus aplikasi. ViewModel memiliki peran

utama sebagai penyimpanan logika bisnis dan data aplikasi yang digunakan oleh antarmuka pengguna. dengan menggunakan ini, maka data seperti daftar karakter tidak akan hilang ketika aktivitas diubah, misalnya seperti orientasi layar. Contoh implementasi ViewModel dalam kode saya pada bagian ini:

CharacterViewModel.kt

```
class CharacterViewModel : ViewModel() {

    private val _characterList = MutableStateFlow<List<Character>>(emptyList())
    val characterList: StateFlow<List<Character>> = _characterList
```

Gambar 14. Screenshot Hasil Jawaban Soal 1 Modul 4

b. Menggunakan ViewModelFactory dalam pembuatan ViewModel

File ini menyediakan cara khusus untuk membuat CharacterViewModel, terutama jika ingin memebri argument ke ViewModel nanti, semisal repository atau parameter lainnya. File ini diperlukan karena viewModel() di Compose memerlukan Factory data ViewModel memiliki non-default. Pembuatan file ViewModelFactory ini agar kode lebih mudah dikembangkan dan dikelola ketika aplikasi akan bertambah besar dan semakin kompleks. Contoh implementasi kode ViewModelFactory saya adalah sebagai berikut:

CharacterViewModelFactory.kt

```
class CharacterViewModelFactory : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(CharacterViewModel::class.java)) {
            return CharacterViewModel() as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Gambar 15. Screenshot Hasil Jawaban Soal 1 Modul 4

c. Menggunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment

StateFlow digunakan untuk menyimpan state secara reaktif. Jadi, ketika data berubah, maka UI akan otomatis terupdate. Contoh implementasi StateFlow di kode saya adalah sebagai berikut:

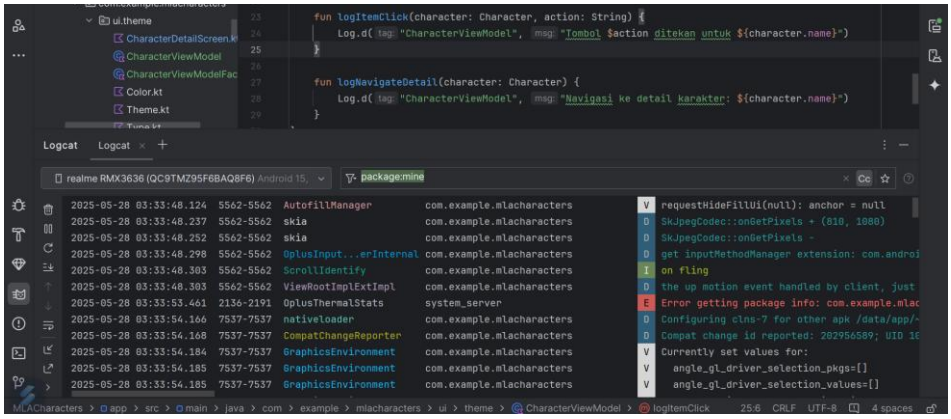
CharacterViewModel.kt

```
13      private val _characterList = MutableStateFlow<List<Character>>(emptyList())
14      val characterList: StateFlow<List<Character>> = _characterList
```

Gambar 16. Screenshot Hasil Jawaban Soal 1 Modul 4

d. Menggunakan logging untuk event:

Tampilan pada Logcat:



Gambar 17. Screenshot Hasil Jawaban Soal 1 Modul 4

- Log saat data item masuk ke dalam list

  Log ini akan tercetak saat setCharacterList() dipanggil—biasanya setelah kamu memuat data karakter. Tujuannya untuk memastikan bahwa list karakter berhasil dimasukkan ke dalam StateFlow.

  CharacterViewModel.kt

```
18          Log.d( tag: "CharacterViewModel", msg: "Data item masuk ke dalam list")
```

Gambar 18. Screenshot Hasil Jawaban Soal 1 Modul 4

- Log saat tombol Detail dan tombol Explicit Intent ditekan

  Fungsi ini dipanggil saat user menekan tombol Detail atau tombol Explicit Intent. Parameternya adalah nama karakter dan jenis tombol yang ditekan.

  CharacterViewModel.kt

```
23      fun logItemClick(character: Character, action: String) {
24          Log.d( tag: "CharacterViewModel", msg: "Tombol $action ditekan untuk ${character.name}")
25      }
```

Gambar 19. Screenshot Hasil Jawaban Soal 1 Modul 4

- Log data dari list yang dipilih ketika berpindah ke halaman Detail

  Fungsi ini dicetak saat user memilih karakter untuk melihat halaman detail. Ini penting karena untuk tahu karakter mana yang dikirim saat navigasi terjadi.

  CharacterViewModel.kt

```
fun logNavigateDetail(character: Character) {
    Log.d( tag: "CharacterViewModel",  msg: "Navigasi ke detail karakter: ${character.name}")
}
```

Gambar 20. Screenshot Hasil Jawaban Soal 1 Modul 4

e. Menggunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi dengan menggunakan breakpoint dan menggunakan fitur Step Over, Step Over, dan Step Out

- Debugging dengan beberapa breakpoint

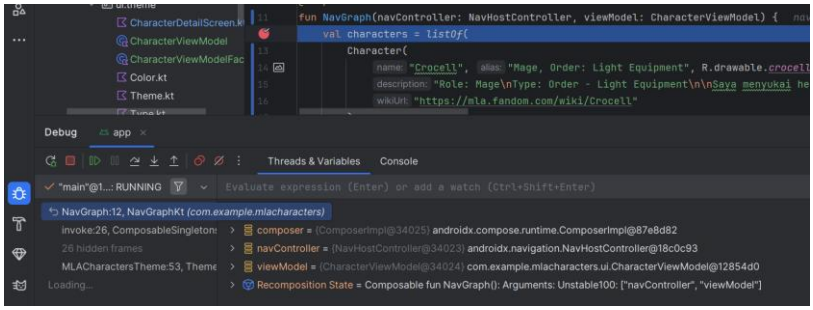
  Fungsi debugger adalah alat untuk membantu developer menemukan dan memperbaiki kesalahan (bug) dlam kode program dengan cara menjalankan program secara perlahan-lahan dan memeriksa apa yang terjadi di setiap langkahnya. Fungsi debugger adalah:

  1. Menghentikan eksekusi program di titik tertentu menggunakan breakpoint.
  2. Memeriksa nilai variabel saat program berhenti.
  3. Menjalankan kode baris per baris agar bisa melihat alur program dan menemukan bug.
  4. Memeriksa call stack, kondisi memori, dan status thread.
  5. Mengubah nilai variabel saat debugging untuk menguji hasilnya.

  Cara menggunakan debugger adalah sebagai berikut:

  1. Pasang breakpoint (terletak di sebelah kiri kode, di angka baris nomor) pada baris yang ingin diperiksa.
  2. Jalankan aplikasi dengan icon kumbang (bug), jangan run biasa.
  3. Jalankan aplikasi di emulator atau device. Saya memakai device agar lebih cepat.
  4. Memeriksa variabel dan status.

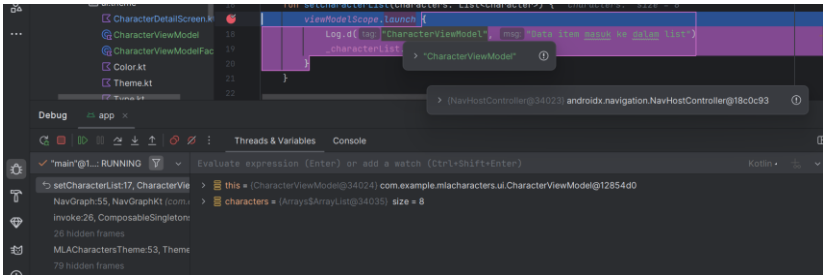
5. Melanjutkan eksekusi setelah berhenti di breakpoint.



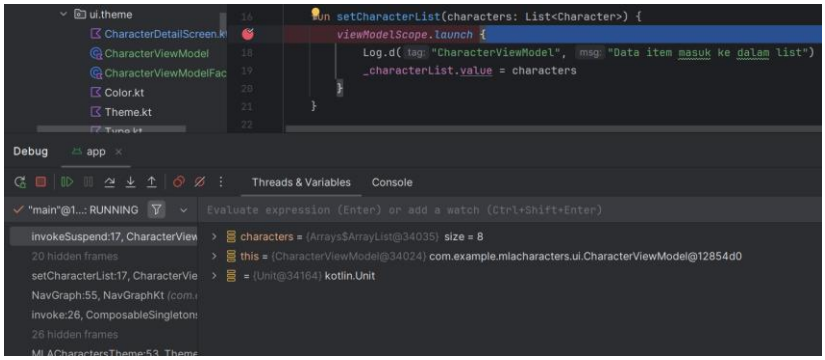
Gambar 21. Screenshot Hasil Jawaban Soal 1 Modul 4

- Menggunakan fitur Step Into (fn + F7)

  Digunakan untuk melangkah masuk ke dalam fungsi (method) yang sedang dipanggil di baris tersebut agar developer dapat melihat bagaimana isi fungsi itu bekerja secara detail.



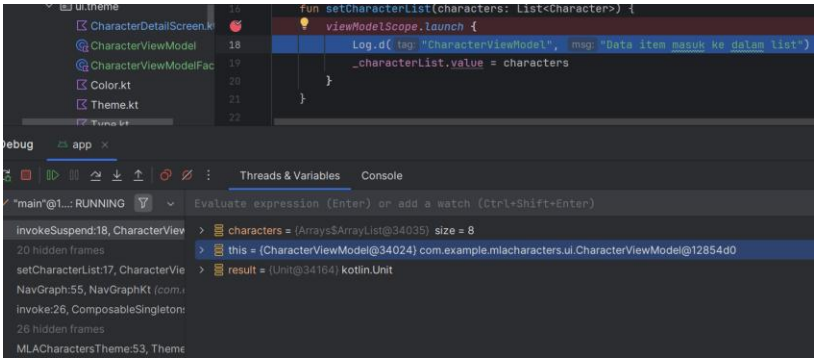
Gambar 22. Screenshot Hasil Jawaban Soal 1 Modul 4



Gambar 23. Screenshot Hasil Jawaban Soal 1 Modul 4

- Menggunakan fitur Step Over (fn + F8)

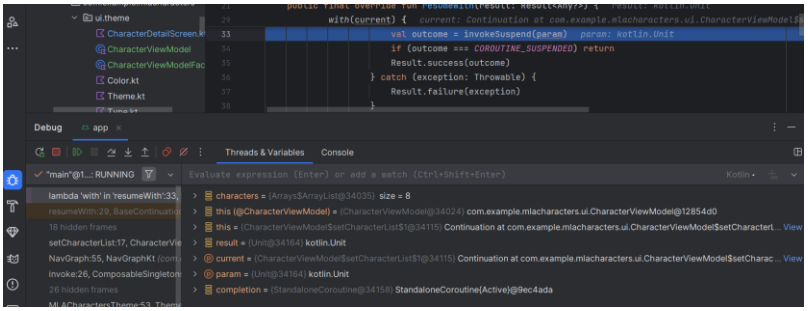
  Digunakan untuk melangkah ke baris berikutnya di fungsi yang sama tanpa masuk ke fungsi lain, gunanya agar developer dapat melompat ke perintah berikutnya tanpa melihat isi fungsi yang dipanggil.



Gambar 24. Screenshot Hasil Jawaban Soal 1 Modul 4

- Menggunakan fitur Step Out (fn + Shift + F8)

  Digunakan untuk melanjutkan eksekusi sampai keluar dari fungsi yang sedang developer jalani sekarang, gunanya agar jika developer sudah masuk terlalu dalam ke fungsi dan ingin langsung keluar ke fungsi pemanggil.



Gambar 25. Screenshot Hasil Jawaban Soal 1 Modul 4

2. Fungsi Application class dalam arsitektur aplikasi Android

   Dalam Application arsitektur Android, Application class adaah komponen inti yang mempresentasikan seluruh siklus hidup aplikasi. Class ini merupakan turunan dari android.app.Application dan bertindak sebagai entry point global untuk menjaga status dan konfigurasi aplikasi yang bersifat global selama aplikasi berjalan. Fungsi-fungsinya adalah sebagai berikut:

   a. Tempat menyimpan pengaturan global

      Jika memiliki data atau pengaturan yang dibutuhkan di banyak bagian aplikasi, misalnya status login, tema, atau bahasa), developer bisa menyimpan di class ini agar tidak perlu mengatur ulang di setiap layar.

b. Inisialisasi satu kali di awal aplikasi

Jika developer memakai library atau tools tambahan (misalnya database, Firebase, atau logger), bisa meletakkan kodenya di class ini agar langsung aktif saat aplikasi dibuka.

c. Memiliki context global

Class ini bisa memberikan konteks yang bisa diakses dari mana saja di aplikasi karena terkadang developer membutuhkan konteks yang tidak bergantung pada layar tertentu

d. Melacak aktivitas aplikasi

Bisa memantau aplikasi seperti kapan layar dibuka atau ditutup. Cocok untuk aplikasi besar yang perlu log atau analytics.

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. ThemePreferenceManager/datastore/data

```
1  package com.example.mladventure.data.datastore
2
3  import android.content.Context
4  import
5  androidx.datastore.preferences.core.booleanPreferencesK
6  ey
7  import androidx.datastore.preferences.core.edit
8  import
9  androidx.datastore.preferences.preferencesDataStore
10 import kotlinx.coroutines.flow.Flow
11 import kotlinx.coroutines.flow.map
12
13 private         val         Context.dataStore         by
14 preferencesDataStore(name = "settings")
15
16 class   ThemePreferenceManager(private   val   context:
17 Context) {
18     companion object {
19         private        val        DARK_MODE_KEY        =
20 booleanPreferencesKey("dark_mode")
21     }
22
23     val        isDarkMode:        Flow<Boolean>        =
24 context.dataStore.data.map { prefs ->
25         prefs[DARK_MODE_KEY] ?: false
26     }
27
28     suspend fun setDarkMode(enabled: Boolean) {
29         context.dataStore.edit { prefs ->
30             prefs[DARK_MODE_KEY] = enabled
31         }
```

```
32      }
33  }
```

Tabel 17. Source Code Soal 1 Modul 5

2. CharacterDao/local/data

```
1   package com.example.mladventure.data.local
2
3   import androidx.room.*
4   import kotlinx.coroutines.flow.Flow
5
6   @Dao
7   interface CharacterDao {
8       @Query("SELECT * FROM characters")
9       fun getAll(): Flow<List<CharacterEntity>>
10
11      @Insert(onConflict = OnConflictStrategy.REPLACE)
12      suspend fun insertAll(characters:
13  List<CharacterEntity>)
14
15      @Update
16      suspend fun update(character: CharacterEntity)
17  }
```

Tabel 18. Source Code Soal 1 Modul 5

3. CharacterDatabase/local/data

```
1  package com.example.mladventure.data.local
2
3  import android.content.Context
4  import androidx.room.Database
5  import androidx.room.Room
6  import androidx.room.RoomDatabase
7
8  @Database(entities = [CharacterEntity::class], version =
9  1)
10 abstract class CharacterDatabase : RoomDatabase() {
11     abstract fun characterDao(): CharacterDao
12
13     companion object {
14         @Volatile      private      var      INSTANCE:
15 CharacterDatabase? = null
16
17         fun        getInstance(context:        Context):
18 CharacterDatabase =
19             INSTANCE ?: synchronized(this) {
20                 Room.databaseBuilder(
21                     context.applicationContext,
22                     CharacterDatabase::class.java,
23                     "character_database"
24                 ).build().also { INSTANCE = it }
25             }
26     }
27 }
```

Tabel 19. Source Code Soal 1 Modul 5

4. CharacterEntity/local/data

```
1  package com.example.mladventure.data.local
2
3  import androidx.room.Entity
4  import androidx.room.PrimaryKey
5
6  @Entity(tableName = "characters")
7  data class CharacterEntity(
8      @PrimaryKey val name: String,
9      val alias: String,
10     val imageUrl: String,
11     val description: String,
12     val wikiUrl: String,
13     val isFavorite: Boolean = false
14 )
```

Tabel 20. Source Code Soal 1 Modul 5

5. ApiResponse/remote/data

```
1  package com.example.mladventure.data.remote
2
3  sealed class ApiResponse<out T> {
4      data class Success<T>(val data: T) : ApiResponse<T>()
5      data class Error(val message: String) :
6  ApiResponse<Nothing>()
7      object Loading : ApiResponse<Nothing>()
8  }
```

Tabel 21. Source Code Soal 1 Modul 5

6. CharacterApi/remote/data

```
1  package com.example.mladventure.data.remote
2
3  import retrofit2.http.GET
4
5  interface CharacterApi {
6      @GET("characters")
7      suspend fun getCharacters(): List<CharacterDto>
8
9      companion object {
10         const          val          BASE_URL          =
11 "https://mlacharacters.free.beeceptor.com/data"
12     }
13 }
```

Tabel 22. Source Code Soal 1 Modul 5

7. CharacterDto/remote/data

```
1  package com.example.mladventure.data.remote
2
3  import kotlinx.serialization.Serializable
4
5  @Serializable
6  data class CharacterDto(
7      val name: String,
8      val alias: String,
9      val imageUrl: String,
10     val description: String,
11     val wikiUrl: String
12 )
```

Tabel 23. Source Code Soal 1 Modul 5

8. CharacterRepository/repository

```
1  package com.example.mladventure.repository
2
3  import com.example.mladventure.data.local.CharacterDao
4  import
5  com.example.mladventure.data.local.CharacterEntity
6  import com.example.mladventure.data.remote.CharacterApi
7  import kotlinx.coroutines.flow.Flow
8
9  class CharacterRepository(
10     private val api: CharacterApi,
11     private val dao: CharacterDao
12 ) {
13     val   characters:   Flow<List<CharacterEntity>>   =
14 dao.getAll()
15
16     suspend fun refreshCharacters() {
17         try {
18             val dtoList = api.getCharacters()
19             val entities = dtoList.map {
20                 CharacterEntity(
21                     name = it.name,
22                     alias = it.alias,
23                     imageUrl = it.imageUrl,
24                     description = it.description,
25                     wikiUrl = it.wikiUrl
26                 )
27             }
28             dao.insertAll(entities)
29         } catch (e: Exception) {
30             e.printStackTrace()
31         }
```

```
32      }
33
34      suspend         fun         toggleFavorite(character:
35  CharacterEntity) {
36          dao.update(character.copy(isFavorite           =
37  !character.isFavorite))
38      }
39  }
```

Tabel 24. Source Code Soal 1 Modul 5

9. SettingScreen.kt/settings

```
1  package com.example.mladventure.settings
2
3  import androidx.compose.foundation.layout.*
4  import androidx.compose.material3.*
5  import androidx.compose.runtime.Composable
6  import androidx.compose.ui.Alignment
7  import androidx.compose.ui.Modifier
8  import androidx.compose.ui.unit.dp
9
10 @Composable
11 fun SettingsScreen(
12     isDarkMode: Boolean,
13     onToggleTheme: (Boolean) -> Unit
14 ) {
15     Column(
16         modifier = Modifier
17             .fillMaxSize()
18             .padding(24.dp),
19         verticalArrangement = Arrangement.Center,
20         horizontalAlignment                                =
21 Alignment.CenterHorizontally
22     ) {
23         Text("Dark         Mode",         style         =
24 MaterialTheme.typography.titleMedium)
25         Spacer(modifier = Modifier.height(8.dp))
26         Switch(
27             checked = isDarkMode,
28             onCheckedChange = onToggleTheme
29         )
30     }
31 }
```

Tabel 25. Source Code Soal 1 Modul 5

10. CharacterDetailScreen.kt/ui.theme

```
1   package com.example.mladventure.ui.theme
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.lazy.LazyColumn
6   import androidx.compose.material.icons.Icons
7   import androidx.compose.material.icons.filled.ArrowBack
8   import androidx.compose.material3.*
9   import androidx.compose.runtime.Composable
10  import androidx.compose.ui.Modifier
11  import androidx.compose.ui.graphics.Color
12  import androidx.compose.ui.layout.ContentScale
13  import androidx.compose.ui.unit.dp
14  import androidx.compose.ui.unit.sp
15  import androidx.navigation.NavController
16  import coil.compose.rememberAsyncImagePainter
17  import
18  com.example.mladventure.data.local.CharacterEntity
19
20
21  @OptIn(ExperimentalMaterial3Api::class)
22  @Composable
23  fun  CharacterDetailScreen(character:  CharacterEntity,
24  navController: NavController) {
25      Scaffold(
26          containerColor = Color(0xFF1C1C1E),
27          topBar = {
28              CenterAlignedTopAppBar(
29                  title = {},
30                  navigationIcon = {
31
```

```
32                      IconButton(onClick       =       {
33  navController.popBackStack() }) {
34                          Icon(
35                              imageVector              =
36  Icons.Default.ArrowBack,
37                              contentDescription       =
38  "Back",
39                              tint = Color.White
40                          )
41                      }
42                  },
43                  colors                                =
44  TopAppBarDefaults.centerAlignedTopAppBarColors(
45                      containerColor = Color.Transparent
46                  )
47              )
48          }
49      ) { innerPadding ->
50          LazyColumn(
51              modifier = Modifier
52                  .padding(innerPadding)
53                  .fillMaxSize()
54          ) {
55              item {
56                  Image(
57                      painter                           =
58  rememberAsyncImagePainter(character.imageUrl),
59                      contentDescription                =
60  character.name,
61                      contentScale = ContentScale.Crop,
62                      modifier = Modifier
```

```
63                          .fillMaxWidth()
64                          .height(500.dp)
65                  )
66              }
67
68              item {
69                  Column(modifier                          =
70  Modifier.padding(16.dp)) {
71                      Row(
72                          modifier                         =
73  Modifier.fillMaxWidth(),
74                          horizontalArrangement            =
75  Arrangement.SpaceBetween
76                      ) {
77                          Text(
78                              text = character.name,
79                              color = Color.White,
80                              style                        =
81  MaterialTheme.typography.titleLarge
82                          )
83                          Text(
84                              text = "2016",
85                              color = Color.LightGray,
86                              style                        =
87  MaterialTheme.typography.bodyMedium
88                          )
89                      }
90
91                      Spacer(modifier                      =
92  Modifier.height(8.dp))
93
```

```
94                      Text(
95                          text = "Deskripsi:",
96                          color = Color.White,
97                          fontWeight                    =
98  androidx.compose.ui.text.font.FontWeight.Bold,
99                          style                         =
100 MaterialTheme.typography.bodyMedium
101                     )
102
103                     Spacer(modifier                   =
104 Modifier.height(4.dp))
105
106                     Text(
107                         text = character.description,
108                         color = Color(0xFFCCCCCC),
109                         style                         =
110 MaterialTheme.typography.bodySmall,
111                         lineHeight = 20.sp
112                     )
113                 }
114             }
115         }
116     }
117 }
118
```

Tabel 26. Source Code Soal 1 Modul 5

11. CharacterViewModel/ui.theme

```
1   package com.example.mladventure.ui.theme
2
3   import android.util.Log
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.viewModelScope
6   import com.example.mladventure.Character
7   import kotlinx.coroutines.flow.MutableStateFlow
8   import kotlinx.coroutines.flow.StateFlow
9   import kotlinx.coroutines.launch
10
11  class CharacterViewModel : ViewModel() {
12
13      private        val        _characterList        =
14  MutableStateFlow<List<Character>>(emptyList())
15      val  characterList:  StateFlow<List<Character>>  =
16  _characterList
17
18      fun setCharacterList(characters: List<Character>) {
19          viewModelScope.launch {
20              Log.d("CharacterViewModel",   "Data   item
21  masuk ke dalam list")
22              _characterList.value = characters
23          }
24      }
25
26      fun  logItemClick(character:  Character,  action:
27  String) {
28          Log.d("CharacterViewModel",   "Tombol   $action
29  ditekan untuk ${character.name}")
30      }
31
```

```
32      fun logNavigateDetail(character: Character) {
33          Log.d("CharacterViewModel", "Navigasi ke detail
34  karakter: ${character.name}")
35      }
36  }
```

Tabel 27. Source Code Soal 1 Modul 5

12. CharacterViewModelFactory/ui.theme

```
1   package com.example.mladventure.ui.theme
2
3
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6
7   class CharacterViewModelFactory :
8   ViewModelProvider.Factory {
9       override fun <T : ViewModel> create(modelClass:
10  Class<T>): T {
11          if
12  (modelClass.isAssignableFrom(CharacterViewModel::class
13  .java)) {
14              return CharacterViewModel() as T
15          }
16          throw IllegalArgumentException("Unknown
17  ViewModel class")
18      }
19  }
```

Tabel 28. Source Code Soal 1 Modul 5

13. Character

```
1  package com.example.mladventure
2
3  data class Character(
4      val name: String,
5      val alias: String,
6      val imageRes: Int,
7      val description: String,
8      val wikiUrl: String
9  )
```

Tabel 29. Source Code Soal 1 Modul 5

14. CharacterListScreen.kt

```
1   package com.example.mladventure
2
3   import android.content.Intent
4   import android.net.Uri
5   import androidx.compose.foundation.Image
6   import androidx.compose.foundation.layout.*
7   import androidx.compose.foundation.lazy.LazyColumn
8   import androidx.compose.foundation.lazy.items
9   import
10  androidx.compose.foundation.shape.RoundedCornerShape
11  import androidx.compose.material3.*
12  import androidx.compose.runtime.Composable
13  import androidx.compose.runtime.collectAsState
14  import androidx.compose.runtime.getValue
15  import androidx.compose.ui.Alignment
16  import androidx.compose.ui.Modifier
17  import androidx.compose.ui.graphics.Color
18  import androidx.compose.ui.layout.ContentScale
19  import androidx.compose.ui.platform.LocalContext
20  import androidx.compose.ui.res.painterResource
21  import androidx.compose.ui.text.font.FontWeight
22  import androidx.compose.ui.unit.dp
23  import androidx.navigation.NavController
24  import
25  com.example.mladventure.ui.theme.CharacterViewModel
26
27  @OptIn(ExperimentalMaterial3Api::class)
28  @Composable
29  fun CharacterListScreen(
30      navController: NavController,
31      viewModel: CharacterViewModel
```

```
) {
    val characters by
viewModel.characterList.collectAsState()
    val context = LocalContext.current


    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(8.dp),
        contentPadding = PaddingValues(top = 60.dp),
        verticalArrangement =
Arrangement.spacedBy(12.dp)
    ) {
        items(characters) { character ->
            Card(
                shape = RoundedCornerShape(12.dp),
                colors =
CardDefaults.cardColors(containerColor =
Color(0xFF2B2B2B)),
                elevation =
CardDefaults.cardElevation(4.dp),
                modifier = Modifier.fillMaxWidth()
            ) {
                Row(modifier = Modifier.padding(12.dp))
{
                    Image(
                        painter = painterResource(id =
character.imageRes),
                        contentDescription =
character.name,
```

```
63                          contentScale                    =
64  ContentScale.Crop,
65                          modifier = Modifier
66                              .width(90.dp)
67                              .height(140.dp)
68                              .padding(end = 12.dp)
69                      )
70
71                      Column(modifier                     =
72  Modifier.weight(1f)) {
73                          Row(
74                              modifier                    =
75  Modifier.fillMaxWidth(),
76                              horizontalArrangement       =
77  Arrangement.SpaceBetween
78                          ) {
79                              Text(
80                                  text = character.name,
81                                  style                   =
82  MaterialTheme.typography.titleMedium,
83                                  color = Color.White,
84                                  fontWeight              =
85  FontWeight.Bold
86                              )
87                              Text(
88                                  text = "2016", // Masih
89  dummy, bisa pakai data jika tersedia
90                                  style                   =
91  MaterialTheme.typography.labelMedium,
92                                  color = Color.LightGray
93                              )
```

```
94                          }
95
96                          Spacer(modifier                =
97  Modifier.height(6.dp))
98
99                          Row(
100                             modifier                   =
101 Modifier.fillMaxWidth(),
102                             verticalAlignment          =
103 Alignment.Top
104                         ) {
105                             Text(
106                                 text = "Deskripsi: ",
107                                 color = Color.White,
108                                 fontWeight             =
109 FontWeight.Bold,
110                                 style                  =
111 MaterialTheme.typography.bodyMedium
112                             )
113
114                             Spacer(modifier            =
115 Modifier.width(8.dp))
116
117                             Text(
118                                 text                   =
119 character.description,
120                                 color = Color.White,
121                                 style                  =
122 MaterialTheme.typography.bodySmall,
123                                 modifier               =
124 Modifier.weight(1f)
```

```
125                                 )
126                             }
127
128                             Spacer(modifier                 =
129 Modifier.height(10.dp))
130
131                             Row(horizontalArrangement       =
132 Arrangement.spacedBy(8.dp)) {
133                                 Button(
134                                     onClick = {
135
136 viewModel.logItemClick(character, "Detail Hero")
137                                         val     intent      =
138 Intent(Intent.ACTION_VIEW,
139 Uri.parse(character.wikiUrl))
140
141 context.startActivity(intent)
142                                     },
143                                     colors                  =
144 ButtonDefaults.buttonColors(containerColor                  =
145 Color(0xFF9BB1EB)),
146                                     modifier                =
147 Modifier.weight(1f)
148                                 ) {
149                                     Text("Detail     Hero",
150 color = Color.White)
151                                 }
152
153                                 Button(
154                                     onClick = {
155
```

```
156
157 viewModel.logItemClick(character, "Deskripsi")
158
159 navController.navigate("detail/${character.name}")
160                                     },
161                                     colors                =
162 ButtonDefaults.buttonColors(containerColor            =
163 Color(0xFF9BB1EB)),
164                                     modifier              =
165 Modifier.weight(1f)
166                                 ) {
167                                     Text("Deskripsi", color
168 = Color.White)
169                                 }
170                             }
171                         }
172                     }
173                 }
174             }
175     }
176 }
```

Tabel 30. Source Code Soal 1 Modul 5

15. MainActivity

```
1  package com.example.mladventure
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.compose.material3.MaterialTheme
7  import androidx.compose.material3.Surface
8  import androidx.lifecycle.viewmodel.compose.viewModel
9  import
10 androidx.navigation.compose.rememberNavController
11 import
12 com.example.mladventure.ui.theme.CharacterViewModel
13 import
14 com.example.mladventure.ui.theme.CharacterViewModelFac
15 tory
16 import
17 com.example.mladventure.ui.theme.MLACharactersTheme
18 import
19 com.example.mladventure.data.datastore.ThemePreference
20 Manager
21 import kotlinx.coroutines.flow.first
22 import androidx.compose.runtime.*
23 import kotlinx.coroutines.runBlocking
24
25 class MainActivity : ComponentActivity() {
26     private      lateinit      var      themeManager:
27 ThemePreferenceManager
28
29     override fun onCreate(savedInstanceState: Bundle?)
30 {
31         super.onCreate(savedInstanceState)
```

```
32          themeManager                                        =
33  ThemePreferenceManager(applicationContext)
34
35          setContent {
36              val                isDarkMode                by
37  themeManager.isDarkMode.collectAsState(initial = false)
38
39              MLACharactersTheme(darkTheme  =  isDarkMode)
40  {
41                  Surface(color                               =
42  MaterialTheme.colorScheme.background) {
43                      val          navController          =
44  rememberNavController()
45                      val viewModel: CharacterViewModel =
46  viewModel(
47                          factory                             =
48  CharacterViewModelFactory()
49                      )
50
51                      NavGraph(
52                          navController = navController,
53                          viewModel = viewModel,
54                          isDarkMode = isDarkMode,
55                          onToggleTheme = { enabled ->
56                              runBlocking {
57
58  themeManager.setDarkMode(enabled)
59                              }
60                          }
61                      )
62                  }
```

```
63                  }
64              }
65          }
66      }
67
```

Tabel 31. Source Code Soal 1 Modul 5

16. MLACharacterApplication.kt

```
1  package com.example.mladventure
2
3  import kotlinx.serialization.json.Json
4  import okhttp3.MediaType
5  import okhttp3.MediaType.Companion.toMediaType
6  import okhttp3.RequestBody
7  import okhttp3.ResponseBody
8  import retrofit2.Converter
9  import retrofit2.Retrofit
10 import java.lang.reflect.Type
11 import kotlinx.serialization.serializer
12 import kotlinx.serialization.encodeToString
13 import kotlinx.serialization.decodeFromString
14 import okhttp3.RequestBody.Companion.toRequestBody
15
16 class JsonConverterFactory(
17     private val json: Json,
18     private val contentType: MediaType
19 ) : Converter.Factory() {
20
21     companion object {
22         fun create(
23             json: Json = Json { ignoreUnknownKeys = true
24 },
25             contentType:          MediaType          =
26 "application/json".toMediaType()
27         ):            JsonConverterFactory           =
28 JsonConverterFactory(json, contentType)
29     }
30
31     override fun responseBodyConverter(
```

```
        type: Type,
        annotations: Array<Annotation>,
        retrofit: Retrofit
    ): Converter<ResponseBody, *> {
        val                 serializer                =
json.serializersModule.serializer(type)
        return Converter { body ->
            json.decodeFromString(serializer,
body.string())
        }
    }

    override fun requestBodyConverter(
        type: Type,
        parameterAnnotations: Array<Annotation>,
        methodAnnotations: Array<Annotation>,
        retrofit: Retrofit
    ): Converter<*, RequestBody> {
        val                 serializer                =
json.serializersModule.serializer(type)
        return Converter<Any, RequestBody> { value ->
            val                 content                =
json.encodeToString(serializer, value)
            content.toRequestBody(contentType)
        }
    }
}
```

Tabel 32. Source Code Soal 1 Modul 5

17. NavGraph.kt

```
1   package com.example.mladventure
2
3   import androidx.compose.runtime.Composable
4   import androidx.compose.runtime.collectAsState
5   import androidx.navigation.NavHostController
6   import androidx.navigation.compose.NavHost
7   import androidx.navigation.compose.composable
8   import
9   com.example.mladventure.ui.theme.CharacterViewModel
10  import com.example.mladventure.CharacterListScreen
11  import
12  com.example.mladventure.ui.theme.CharacterDetailScreen
13  import com.example.mladventure.settings.SettingsScreen
14
15  @Composable
16  fun NavGraph(
17      navController: NavHostController,
18      viewModel: CharacterViewModel,
19      isDarkMode: Boolean,
20      onToggleTheme: (Boolean) -> Unit
21  ) {
22      val                charactersState                =
23  viewModel.characterList.collectAsState()
24      val characters = charactersState.value
25
26      NavHost(navController       =       navController,
27  startDestination = "list") {
28          composable("list") {
29              CharacterListScreen(
30                  viewModel = viewModel,
31                  navController = navController
```

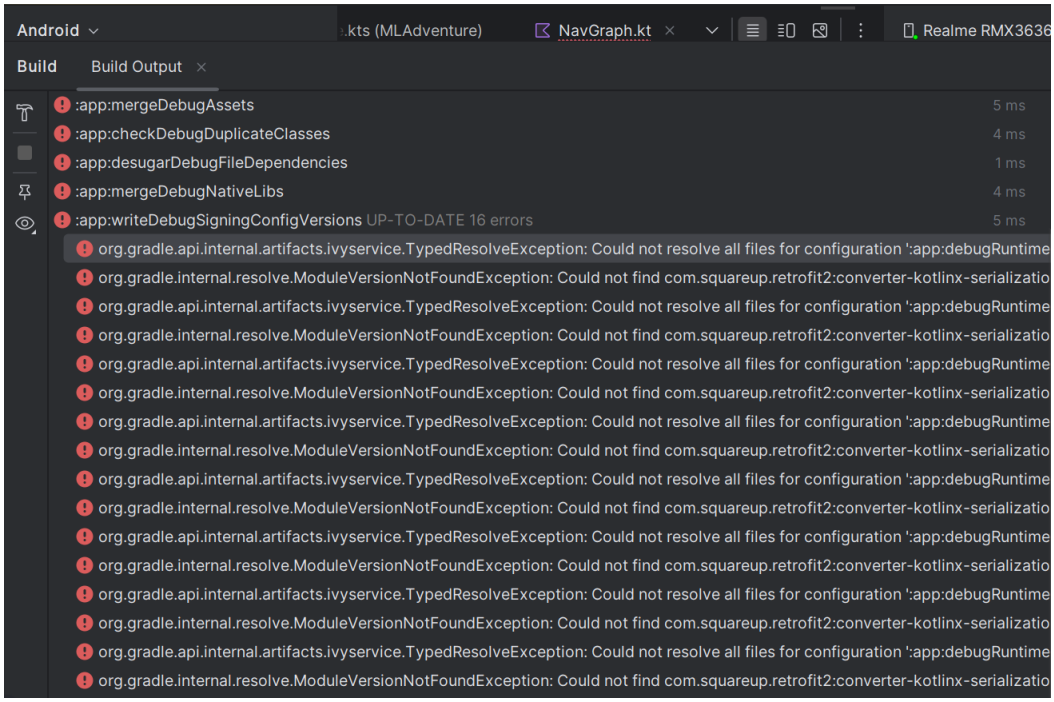
```
            )
        }

        composable("detail/{name}") { backStackEntry ->
            val                    name                    =
backStackEntry.arguments?.getString("name")
            val character = characters.find { it.name ==
name }
            character?.let {
                CharacterDetailScreen(
                    character = it,
                    navController = navController
                )
            }
        }

        composable("settings") {
            SettingsScreen(
                isDarkMode = isDarkMode,
                onToggleTheme = onToggleTheme
            )
        }
    }
}
```

Tabel 33. Source Code Soal 1 Modul 5

B. Output Program



Gambar 26. Screenshot Hasil Jawaban Soal 1 Modul 5

C. Pembahasan

1. ThemePreferenceManager/datastore/data

`ThemePreferenceManager` merupakan kelas yang mengelola penyimpanan preferensi tampilan tema (dark mode) pada aplikasi menggunakan Jetpack DataStore. Dengan memanfaatkan `Flow`, aplikasi dapat merespons perubahan tema secara real-time berdasarkan nilai boolean yang disimpan pada DataStore.

2. CharacterDao/local/data

`CharacterDao` adalah antarmuka yang berfungsi sebagai jembatan antara aplikasi dan database lokal Room. DAO ini menyediakan fungsi untuk mengambil seluruh data karakter, menyimpan daftar karakter, dan memperbarui data karakter, termasuk status favorit, dengan operasi berbasis *suspend* dan *Flow*.

3. CharacterDatabase/local/data

`CharacterDatabase` merupakan kelas abstrak yang memperluas `RoomDatabase` dan bertanggung jawab menginisialisasi instance database Room serta menyediakan akses ke `CharacterDao`.

4. **CharacterEntity/local/data**

   `CharacterEntity` adalah data class yang merepresentasikan struktur tabel dalam Room database dengan anotasi `@Entity`. Kelas ini berisi atribut seperti `name, alias, imageUrl, description, wikiUrl,` dan `isFavorite` yang mewakili data karakter yang disimpan secara lokal.

5. **ApiResponse/remote/data**

   `ApiResponse` adalah kelas sealed yang mendefinisikan status respons dari API. Ada tiga kemungkinan status yaitu `Success, Error`, dan `Loading`. Struktur ini digunakan untuk memudahkan pengelolaan data dan penanganan error saat melakukan permintaan ke API.

6. **CharacterApi/remote/data**

   `CharacterApi` adalah antarmuka Retrofit yang mendeklarasikan endpoint API untuk mengambil data karakter dari internet. Dengan anotasi `@GET`, aplikasi dapat mengambil data berupa daftar karakter (`CharacterDto`) dari endpoint yang telah ditentukan.

7. **CharacterDto/remote/data**

   `CharacterDto` adalah data transfer object (DTO) yang digunakan untuk menyesuaikan struktur data yang diterima dari API. Dengan anotasi `@Serializable`, objek ini bisa dikonversi langsung dari dan ke format JSON menggunakan Kotlinx Serialization.

8. **CharacterRepository/repository**

   `CharacterRepository` bertugas sebagai perantara antara data sumber (API dan database lokal) dan lapisan presentasi (UI).

9. **SettingScreen.kt/settings**

   `SettingScreen.kt` merupakan komponen UI berbasis Jetpack Compose yang menampilkan pengaturan tema.

10. **CharacterDetailScreen.kt/ui.theme**

`CharacterDetailScreen.kt` adalah layar yang menampilkan detail lengkap dari karakter yang dipilih.

11. CharacterViewModel/ui.theme

`CharacterViewModel` merupakan kelas ViewModel yang bertanggung jawab menyimpan state daftar karakter dan menangani log interaksi pengguna. Data disimpan dalam bentuk `StateFlow`, memungkinkan UI untuk memperbarui tampilan secara otomatis saat data berubah.

12. CharacterViewModelFactory/ui.theme

`CharacterViewModelFactory` adalah pabrik ViewModel yang digunakan untuk membuat instance `CharacterViewModel.`

13. Character

`Character` adalah model data biasa (plain Kotlin object) yang digunakan untuk menampilkan data karakter dalam antarmuka pengguna. Objek ini berbeda dari `CharacterEntity` atau `CharacterDto` dan biasanya digunakan di lapisan presentasi agar tidak terikat langsung dengan data dari sumber lokal atau remote.

14. CharacterListScreen.kt

`CharacterListScreen.kt` menampilkan daftar karakter dalam bentuk list.

15. MainActivity

`MainActivity` adalah titik masuk utama aplikasi. Di sini dilakukan pengaturan tema berdasarkan `ThemePreferenceManager`, inisialisasi `ViewModel`, serta pemanggilan fungsi `NavGraph` untuk menavigasi antar layar dalam aplikasi.

16. MLACharacterApplication.kt

`MLACharacterApplication.kt` berisi konfigurasi kustom `JsonConverterFactory` yang digunakan Retrofit untuk mengonversi data JSON menggunakan Kotlinx Serialization.

17. NavGraph.kt

`NavGraph.kt` mengatur semua rute navigasi dalam aplikasi.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

aikoanatashawendiono/Laporan-Praktikum-Pemrograman-Mobile