LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE I

Oleh:

Ghani Mudzakir **NIM. 2310817110011**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE I

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

Nama Praktikan : Ghani Mudzakir
NIM : 2310817110011

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

DAFTAR ISI

DAFTAR GAMBAR

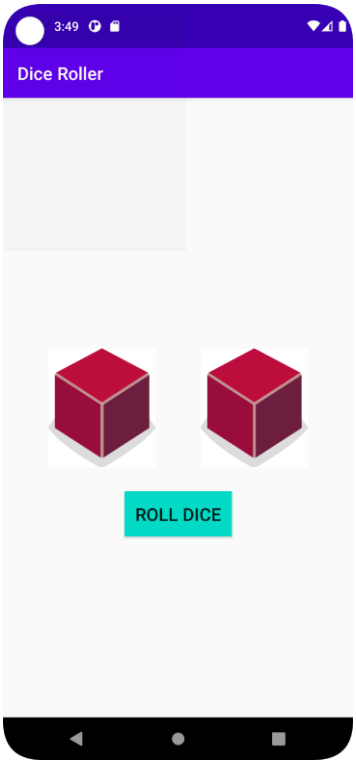
DAFTAR TABEL

MODUL 1 : Android Basic with Kotlin

SOAL 1

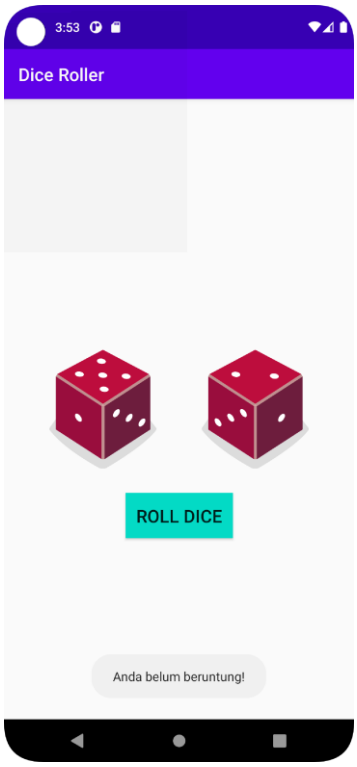
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



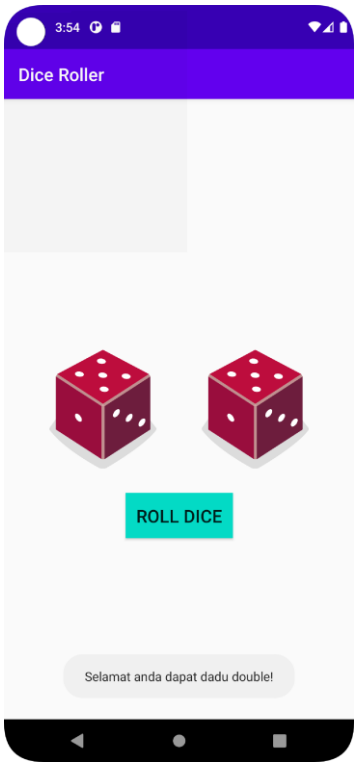
Gambar 1 Tampilan Awal Aplikasi Modul 1

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



Gambar 2 Tampilan Dadu setelah Diroll Modul 1

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



Gambar 3 Tampilan Roll Dadu Double Modul 1

A. Source Code

1. MainActivity.kt

Tabel 1 Source Code MainActivity.kt Jawaban Soal 1 Modul 1

```
1   package com.example.modul_satu_dadu
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.enableEdgeToEdge
6   import androidx.appcompat.app.AppCompatActivity
7   import
8   com.example.modul_satu_dadu.databinding.ActivityMainBind
9   ing
10  import kotlin.random.Random
11
12  class MainActivity : AppCompatActivity() {
13      private lateinit var binding: ActivityMainBinding
14
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          enableEdgeToEdge()
```

```
18
19          binding                                                 =
20  ActivityMainBinding.inflate(layoutInflater)
21          setContentView(binding.root)
22
23          setSupportActionBar(binding.toolbar)
24
25          binding.buttonRollDice.setOnClickListener {
26              rollDice()
27          }
28      }
29
30      private fun getDiceImage(diceRoll: Int): Int {
31          return when (diceRoll) {
32              1 -> R.drawable.dice_1
33              2 -> R.drawable.dice_2
34              3 -> R.drawable.dice_3
35              4 -> R.drawable.dice_4
36              5 -> R.drawable.dice_5
37              6 -> R.drawable.dice_6
38              else -> R.drawable.dice_0
39          }
40      }
41
42      private fun rollDice() {
43          val diceRoll1 = Random.nextInt(1, 7)
44          val diceRoll2 = Random.nextInt(1, 7)
45
46
47  binding.gambarDaduPertama.setImageResource(getDiceImage(
48  diceRoll1))
49
50  binding.gambarDaduKedua.setImageResource(getDiceImage(di
51  ceRoll2))
52
53          val message = if (diceRoll1 == diceRoll2) {
54              "Selamat! Anda mendapatkan dadu double!"
            } else {
                "Anda belum beruntung!"
            }
            Toast.makeText(this,                        message,
    Toast.LENGTH_SHORT).show()
        }
    }
```

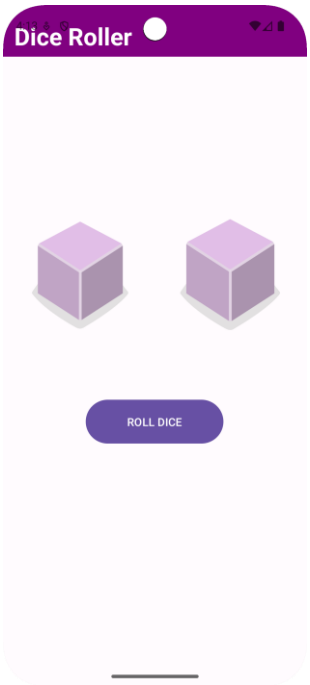
2. activity_main.xml

Tabel 2 Source Code activity_main.xml Jawaban Soal 1

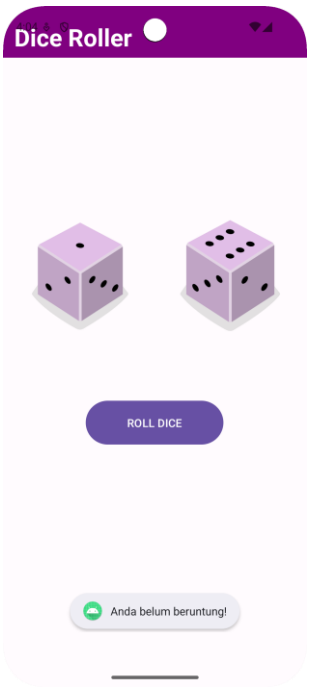
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
3
4   xmlns:android="http://schemas.android.com/apk/res/android
5   "
6       xmlns:app="http://schemas.android.com/apk/res-auto"
7       xmlns:tools="http://schemas.android.com/tools"
8       android:id="@+id/main"
9       android:layout_width="match_parent"
10      android:layout_height="match_parent"
11      tools:context=".MainActivity">
12
13
14      <androidx.appcompat.widget.Toolbar
15          android:id="@+id/toolbar"
16          android:layout_width="match_parent"
17          android:layout_height="wrap_content"
18          android:background="#800080"
19          app:layout_constraintBottom_toBottomOf="parent"
20          app:layout_constraintEnd_toEndOf="parent"
21          app:layout_constraintHorizontal_bias="0.0"
22          app:layout_constraintStart_toStartOf="parent"
23          app:layout_constraintTop_toTopOf="parent"
24          app:layout_constraintVertical_bias="0.0"
25
26  app:popupTheme="@style/ThemeOverlay.AppCompat.Light"
27          app:title=""
28
29  app:titleEnabled="false"></androidx.appcompat.widget.Tool
30  bar>
31
32      <TextView
33          android:id="@+id/toolbar_title"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:text="@string/string_atas"
37          android:textColor="@android:color/white"
38          android:textSize="30sp"
39          android:textStyle="bold"
40          app:layout_constraintBottom_toBottomOf="parent"
41          app:layout_constraintEnd_toEndOf="parent"
42          app:layout_constraintHorizontal_bias="0.06"
43          app:layout_constraintStart_toStartOf="parent"
44          app:layout_constraintTop_toTopOf="parent"
45          app:layout_constraintVertical_bias="0.023" />
46
47      <ImageView
48          android:id="@+id/gambar_dadu_kedua"
```

```
49 50           android:layout_width="180dp"
51              android:layout_height="179dp"
52              android:layout_marginTop="180dp"
53              android:src="@drawable/dice_0"
54              app:layout_constraintEnd_toEndOf="parent"
55              app:layout_constraintHorizontal_bias="0.5"
56
57      app:layout_constraintStart_toEndOf="@id/gambar_dadu_perta
58      ma"
59              app:layout_constraintTop_toBottomOf="@id/toolbar"
60      />
61
62          <ImageView
63              android:id="@+id/gambar_dadu_pertama"
64              android:layout_width="181dp"
65              android:layout_height="173dp"
66              android:layout_marginTop="184dp"
67              android:src="@drawable/dice_0"
68
69      app:layout_constraintEnd_toStartOf="@id/gambar_dadu_kedua
70      "
71              app:layout_constraintHorizontal_bias="0.5"
72              app:layout_constraintStart_toStartOf="parent"
73              app:layout_constraintTop_toBottomOf="@id/toolbar"
74      />
75
76          <Button
77              android:id="@+id/button_roll_dice"
78              android:layout_width="170dp"
79              android:layout_height="62dp"
80              android:text="@string/string_tombol1"
                app:layout_constraintBottom_toBottomOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.497"
                app:layout_constraintStart_toStartOf="parent"

        app:layout_constraintTop_toBottomOf="@id/gambar_dadu_pert
        ama"
                app:layout_constraintVertical_bias="0.178" />

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

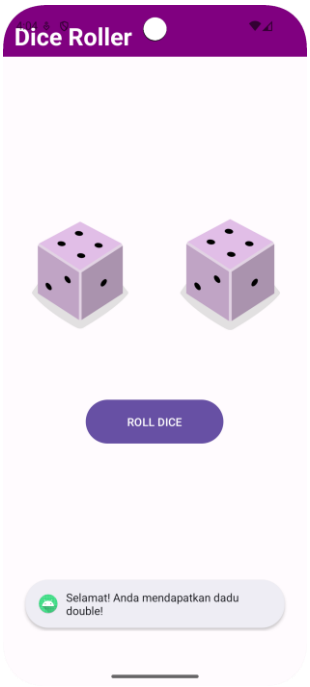
B. Output Program



Gambar 4 Screenshot Hasil Jawaban Soal 1 Modul 1



Gambar 5 Screenshot Hasil Jawaban Soal 1 Saat Dadu Tidak Double Modul 1



Gambar 6 Screenshot Hasil Jawaban Soal 1 Saat Dadu Double Modul 1

C. Pembahasan

1. MainActivity.kt:

Pada line 1, kita mendeklarasikan nama package file Kotlin kita.

Pada line 3-9, kita mengimport beberapa library yang kita gunakan untuk membuat activity, menampilkan pesan notifikasi singkat, menyesuaikan tampilan aplikasi, superclass dari activity, kelas otomatis dari layout activity_main.xml, dan fungsi random untuk mengacak angka.

Pada line 11-25, kita membuat kelas MainActivity sebagai aktigitas utama dan mewarisi dari APPCompatActivity. Kita membuat variabel binding untuk mengakses komponen UI dari layout activity_main.xml, Fungsi onCreate dipanggil ketika activity pertama kali dibuat. Super.onCreate() juga memanggil fungsi milik

AppCompatActivity. Kita juga menambahkan fungsi ketika tombol buttonRollDice ditekan maka fungsi rollDice() akan dijalankan.

Pada line 27-36, kita membuat fungsi getDiceImage() yang mana memiliki parameter integer yang berfungsi akan mengembalikan ID Gambar yang sesuai.

Pada line 38 - 40, kita membuat Fungsi untuk mengacak dan menampilkan hasil dua dadu.

Pada line 42- 46, kita mengatur gambar dadu pertama dan kedua berdasarkan hasil diceRoll1 dan diceRoll2.

Pada line 48 – 46, kita membuat fungsi Toast yang berguna untuk membuat pesan khusus jika dua angka dadu sama (double), dan pesan biasa jika tidak.

2. activity_main.xml

Pada line 1-9, kita membuat layout utama yang membungkus semua elemen yang akan ditampilkan. ConstraintLayout memungkinkan kita mengatur posisi setiap elemen secara fleksibel dengan menetapkan hubungan antar elemen atau terhadap parent layout.

Pada line 12 - 26, kita membuat fungsi toolbar yang berfungsi sebagai action bar (bagian atas aplikasi). Kita menambahkan ID toolbar sehingga bisa kita hubungkan ke Kotlin menggunakan binding.toolbar. Kita menggunakan warna ungu (#800080). Toolbar ini tidak menampilkan judul default karena kita mengatur titleEnabled ke false.

Pada line 28 - 41, kita membuat TextView untuk menampilkan teks sebagai judul toolbar.

Pada line 43 - 63, kita membuat ImageView pertama yang akan menampilkan gambar dadu di sisi kiri layar. Kita awalnya menggunakan gambar dice_0. Elemen ImageView ini kita beri ID gambar_dadu_pertama. Sama seperti gambar_dadu_pertama, kita juga membuat ImageView kedua untukmenampilkan dadu kedua yang tampil di sisi kanan.

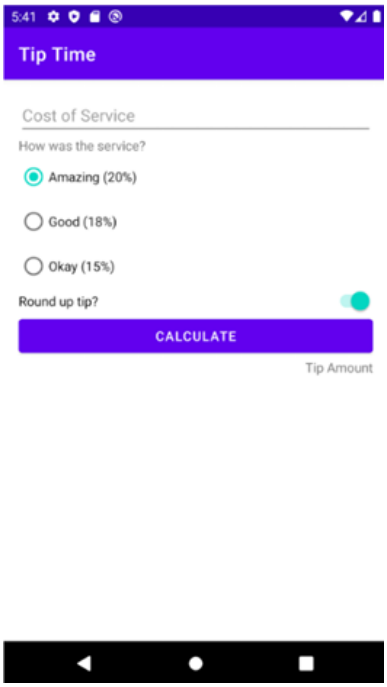
Pada line 67 - 75, kita membuat Button untuk melempar dadu, kita beri teks dari string resource string_tombol1. Saat tombol ini ditekan, fungsi rollDice() di Kotlin dipanggil untuk mengacak angka dan mengganti gambar dadu, serta memunculkan pesan Toast.

MODUL 2 : Android Layout

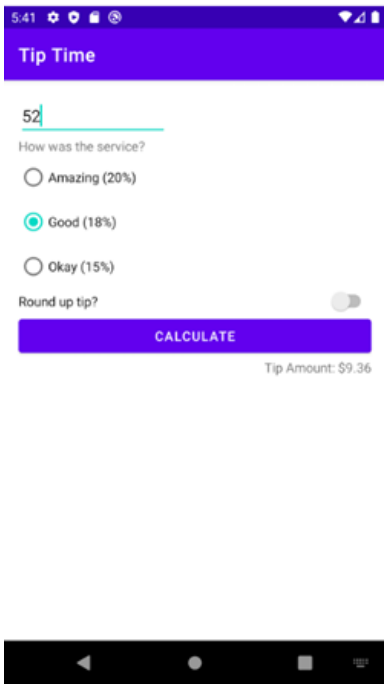
SOAL 1

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

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



Gambar 7 Tampilan Awal Aplikasi Modul 2



Gambar 8 Tampilan Aplikasi Setelah Dijalankan Modul 2

A. Source Code

1. MainActivity.kt

Tabel 3 Source Code MainActivity.kt Jawaban Soal 1 Modul 2

```
1   package com.example.kalkulator_tip
2
3   import android.os.Bundle
4   import android.widget.*
5   import androidx.activity.viewModels
6   import androidx.appcompat.app.AppCompatActivity
7   import
8   com.google.android.material.textfield.TextInputLayout
9   import
10  com.google.android.material.switchmaterial.SwitchMaterial
11  import java.text.NumberFormat
12
13  class MainActivity : AppCompatActivity() {
14
15      private lateinit var costInputLayout: TextInputLayout
16      private lateinit var costOfServiceEditText: EditText
17      private lateinit var tipOptions: RadioGroup
18      private lateinit var roundUpSwitch: SwitchMaterial
19      private lateinit var calculateButton: Button
20      private lateinit var tipResultTextView: TextView
```

```
21
22      private val viewModel: TipViewModel by viewModels()
23
24      override fun onCreate(savedInstanceState: Bundle?) {
25          super.onCreate(savedInstanceState)
26          setContentView(R.layout.activity_main)
27
28          costInputLayout                                    =
findViewById(R.id.cost_input_layout)
29          costOfServiceEditText                              =
findViewById(R.id.cost_of_service)
30          tipOptions = findViewById(R.id.tip_options)
31          roundUpSwitch                                      =
findViewById(R.id.round_up_switch)
32          calculateButton                                    =
findViewById(R.id.calculate_button)
33          tipResultTextView = findViewById(R.id.tip_result)
34
35          if (viewModel.tipAmount.isNotEmpty()) {
36              tipResultTextView.text = viewModel.tipAmount
37          }
38
39          calculateButton.setOnClickListener {
40              calculateTip()
41          }
42      }
43
44      private fun calculateTip() {
45          val                    costText                    =
costOfServiceEditText.text.toString()
46          val cost = costText.toDoubleOrNull()
47
48          if (cost == null || cost <= 0.0) {
49              costInputLayout.error = "Masukkan biaya yang
valid"
50              Toast.makeText(this,   "Input   tidak   valid!
Masukkan angka lebih dari 0", Toast.LENGTH_SHORT).show()
51              tipResultTextView.text = "Tip Amount: -"
52              return
53          } else {
54              costInputLayout.error = null // clear error
55          }
56
57          val          tipPercentage          =          when
(tipOptions.checkedRadioButtonId) {
58              R.id.option_amazing -> 0.20
59              R.id.option_good -> 0.18
60              else -> 0.15
61          }
```

```
66
67          var tip = cost * tipPercentage
68          if (roundUpSwitch.isChecked) {
69              tip = kotlin.math.ceil(tip)
70          }
71
72          val                 formattedTip                 =
NumberFormat.getCurrencyInstance().format(tip)
          val resultText = "Tip Amount: $formattedTip"
          tipResultTextView.text = resultText

          viewModel.tipAmount = resultText
      }
}
```

2. activity_main.xml

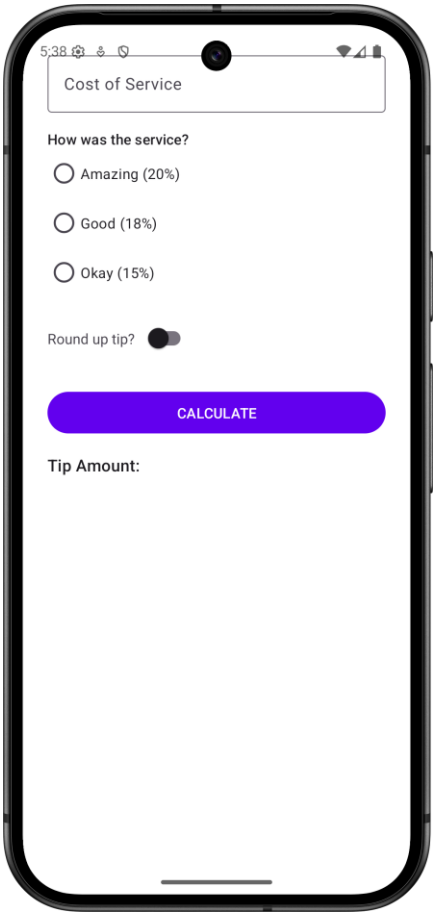
Tabel 4 Source Code activity_main.xml Jawaban Soal 1 Modul 2

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <ScrollView
3   xmlns:android="http://schemas.android.com/apk/res/andr
4   oid"
5       xmlns:app="http://schemas.android.com/apk/res-
6   auto"
7       android:layout_width="match_parent"
8       android:layout_height="match_parent">
9
10      <LinearLayout
11          android:layout_width="match_parent"
12          android:layout_height="wrap_content"
13          android:orientation="vertical"
14          android:padding="24dp"
15
16  android:background="?android:attr/windowBackground">
17
18
19  <com.google.android.material.textfield.TextInputLayout
20              android:id="@+id/cost_input_layout"
21              android:layout_width="match_parent"
22              android:layout_height="wrap_content"
23              android:hint="Cost of Service">
24
25
26  <com.google.android.material.textfield.TextInputEditTe
27  xt
28                  android:id="@+id/cost_of_service"
29                  android:layout_width="match_parent"
30                  android:layout_height="wrap_content"
```

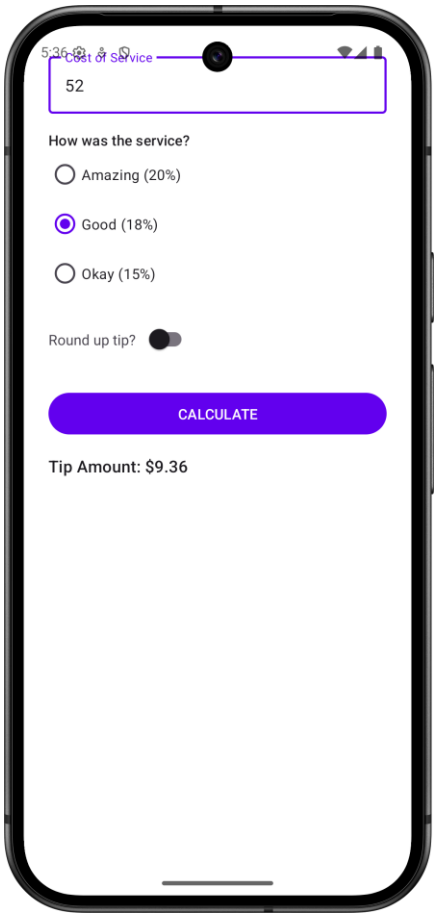
```
                    android:inputType="numberDecimal" />

</com.google.android.material.textfield.TextInputLayou
t>


        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="How was the service?"

android:textAppearance="?attr/textAppearanceTitleSmall
"
            android:layout_marginTop="16dp" />

        <RadioGroup
            android:id="@+id/tip_options"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">


<com.google.android.material.radiobutton.MaterialRadio
Button
                android:id="@+id/option_amazing"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Amazing (20%)" />


<com.google.android.material.radiobutton.MaterialRadio
Button
                android:id="@+id/option_good"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Good (18%)" />


<com.google.android.material.radiobutton.MaterialRadio
Button
                android:id="@+id/option_okay"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Okay (15%)" />

        </RadioGroup>

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

```
81              android:orientation="horizontal"
82              android:layout_marginTop="16dp">
83
84              <TextView
85                  android:layout_width="wrap_content"
86                  android:layout_height="wrap_content"
87                  android:text="Round up tip?"
88
89      android:layout_gravity="center_vertical"/>
90
91
92      <com.google.android.material.switchmaterial.SwitchMate
93      rial
94                  android:id="@+id/round_up_switch"
95                  android:layout_width="wrap_content"
96                  android:layout_height="wrap_content"
97                  android:layout_marginStart="8dp" />
98          </LinearLayout>
99
100
101     <com.google.android.material.button.MaterialButton
102             android:id="@+id/calculate_button"
103             android:layout_width="match_parent"
104             android:layout_height="wrap_content"
105             android:text="CALCULATE"
106             android:layout_marginTop="24dp"/>

          <TextView
              android:id="@+id/tip_result"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Tip Amount:"

      android:textAppearance="?attr/textAppearanceTitleMediu
      m"
              android:layout_marginTop="16dp" />


      </LinearLayout>

      </ScrollView>
```

B. Output Program



Gambar 9 Tampilan Awal Aplikasi Modul 2



Gambar 10 Tampilan Aplikasi Setelah Digunakan Modul 2

C. Pembahasan

1. MainActivity.kt

Pada kode ini kita menuliskan beberapa komponen utama yang akan digunakan pada aplikasi untuk menghitung jumlah tip. Komponen utama yang digunakan seperti EditText untuk input biaya, RadioGroup untuk memilih persentase tip 15%, 18%, atau 20%, SwitchMaterial untuk opsi pembulatan, dan TextView untuk menampilkan hasil. Semua komponen ini dihubungkan dengan layout XML menggunakan fungsi findViewById() yang kita masukan ke variabel yang kita deklarasikan di awal kode. Selain itu, kita juga menggunakan TextInputLayout untuk memberikan peringatan kesalahan yang lebih baik jika terjadi input yang tidak valid.

Fungsi utama terletak pada method calculateTip(), fungsi ini akan dijalankan saat tombol "Calculate" ditekan. Pada method ini, nilai biaya layanan akan diambil dari EditText dan divalidasi terlebih dahulu. Jika input tidak valid seperti kosong atau nilainya kurang dari 0, maka akan ditampilkan pesan error melalui TextInputLayout dan notifikasi Toast. Jika nilai valid, sistem akan menghitung jumlah tip berdasarkan persentase yang dipilih dan menambahkan opsi pembulatan jika switch diaktifkan. Nilai tip yang diperoleh kemudian diformat ke dalam format mata uang menggunakan NumberFormat.getCurrencyInstance() dan ditampilkan di layar melalui TextView. Agar hasil perhitungan tetap tersimpan ketika terjadi perubahan konfigurasi seperti rotasi layar, aplikasi ini menggunakan ViewModel di file TipViewModel.kt yang menyimpan nilai tip terakhir. Sehingga, pengguna tidak perlu menghitung ulang ketika tampilan aplikasi berubah.

2. activity_main.xml

Layout menggunakan ScrollView sebagai komponen utama untuk memungkinkan layar discroll ketika layar perangkat tidak cukup untuk menampilkan semua elemen sekaligus. Di dalam ScrollView, ada sebuah LinearLayout vertikal dengan padding sebesar 24dp yang berfungsi sebagai kontainer utama bagi seluruh elemen UI aplikasi. Penggunaan ?android:attr/windowBackground pada atribut android:background memastikan latar belakang sesuai dengan tema perangkat yang akan digunakan oleh user.

Komponen pertama adalah TextInputLayout yang membungkus TextInputEditText, digunakan untuk menerima input biaya layanan dari pengguna. Komponen input ini diberi hint "Cost of Service" dan memiliki input type numberDecimal agar memungkinkan pengguna untuk memasukkan nilai yang akan dihitung. Setelah itu terdapat TextView sebagai label yang memberikan pertanyaan "How was the service?", diikuti oleh RadioGroup yang berisi tiga pilihan menggunakan MaterialRadioButton, masing-masing untuk persentase tip yang terdiri dari "Amazing (20%)", "Good (18%)", dan "Okay (15%)".

Berikutnya adalah LinearLayout horizontal yang terdiri dari TextView dan SwitchMaterial yang memberikan opsi kepada pengguna apakah ingin membulatkan jumlah tip ke atas atau tidak. Setelah itu terdapat tombol MaterialButton dengan teks "CALCULATE" yang akan memicu proses perhitungan saat ditekan. Terakhir, TextView dengan ID tip_result berfungsi untuk menampilkan hasil akhir dari perhitungan tip. Seluruh komponen disusun secara vertikal dan dibuat agar mudah digunakan, jelas, serta responsif terhadap berbagai ukuran layar perangkat Android.

MODUL 3 : Build a Scrollable

SOAL 1

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

    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)

    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas

    3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah

    4. Terdapat 2 button dalam list, dengan fungsi berikut:

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

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

    5. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius

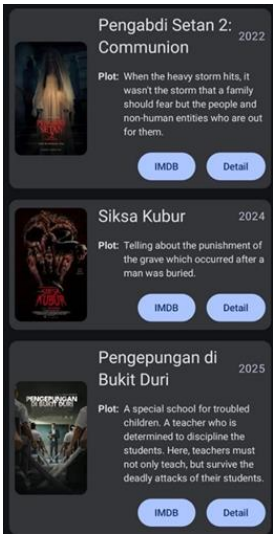
    6. Saat orientasi perangkat berubah/dirotasi, baik ke portrait maupun landscape, aplikasi responsif dan dapat menunjukkan list dengan baik. Data di dalam list tidak boleh hilang

    7. Aplikasi menggunakan arsitektur single activity (satu activity memiliki beberapa fragment)

    8. Aplikasi berbasis XML harus menggunakan ViewBinding

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

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



Gambar 11 Contoh UI List Modul 3

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

Gambar 12 Contoh UI Detail Modul 3

A. Source Code

1. Pembukaan.kt

Tabel 5 Source Code Pembukaan.kt Modul 3

```
1  package com.example.pembukaan_catur
2
3  import androidx.annotation.DrawableRes
4
5  data class TampilanPembukaan(
6      val nama_pembukaan: String,
7      val penjelasan_singkat: String,
8      val penjelasan_dua_paragraf: String,
9      @DrawableRes val imageResId: Int,
10     val link: String
11 )
```

2. PembukaanRepository.kt

Tabel 6 Source Code PembukaanRepository.kt Modul 3

```
1  package com.example.pembukaan_catur
2  sss
3  class PembukaanRepository {
4      fun ambilSemuaPembukaan(): List<TampilanPembukaan>{
5          return listOf(
6              TampilanPembukaan(
7                  nama_pembukaan = "Sicilian Defense",
8                  penjelasan_singkat = "Sebuah pembukaan
9  agresif yang dimulai dengan e4 c5, sering digunakan
10 untuk menciptakan permainan tidak seimbang.",
11                 penjelasan_dua_paragraf = """
12         Sicilian Defense adalah salah satu pembukaan
13 paling populer dan tajam dalam catur modern. Dengan
14 membalas e4 dengan c5, hitam menghindari simetri dan
15 berusaha mengambil inisiatif dengan permainan sayap.
16
17         Pembukaan ini terkenal menghasilkan posisi yang
18 kompleks dan penuh taktik. Banyak juara dunia, seperti
19 Kasparov dan Fischer, mengandalkan Sicilian untuk
20 menghadapi pemain e4.
21     """.trimIndent(),
22                 imageResId = R.drawable.p1,
23                 link                                    =
24 "https://www.chess.com/openings/Sicilian-Defense"
25             ),
26
27             TampilanPembukaan(
```

```
                nama_pembukaan = "French Defense",
                penjelasan_singkat = "Strategi yang
dimulai dengan e4 e6, di mana hitam memblokir pion putih
dan merencanakan serangan melalui posisi yang lebih
tertutup.",
                penjelasan_dua_paragraf = """
        French Defense memberikan struktur pion yang
kokoh dan solid untuk hitam. Ia menawarkan banyak
kemungkinan strategis, terutama dalam permainan tengah.

        Salah satu ciri khasnya adalah konflik antara
pion e5 putih dan struktur pertahanan hitam di d5. Taktik
dan strategi memainkan peran penting dalam membuka
posisi ini.
    """.trimIndent(),
                imageResId = R.drawable.p2,
                link                                    =
"https://www.chess.com/openings/French-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Ruy López Opening",
                penjelasan_singkat = "Pembukaan klasik
yang dimulai dengan e4 e5 2.Nf3 Nc6 3.Bb5, bertujuan
untuk mengontrol pusat dan menekan pertahanan hitam.",
                penjelasan_dua_paragraf = """
        Ruy López adalah salah satu pembukaan tertua
yang masih dimainkan di level tinggi. Dengan menekan
kuda di c6, putih mencoba mengganggu kontrol hitam
terhadap pusat.

        Posisi yang timbul sering bersifat strategis,
dengan ruang untuk manuver jangka panjang dan potensi
serangan raja di tahap akhir pembukaan.
    """.trimIndent(),
                imageResId = R.drawable.p3,
                link                                    =
"https://www.chess.com/openings/Ruy-Lopez-Opening"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Caro-Kann Defense",
                penjelasan_singkat = "Dimulai dengan e4
c6, hitam berusaha untuk memperkuat pusat dan membangun
pertahanan yang solid.",
                penjelasan_dua_paragraf = """
        Caro-Kann terkenal karena stabilitas dan
keamanan bagi raja hitam. Ini adalah pembukaan pilihan
bagi pemain yang menyukai posisi bertahan namun aktif.
```

```
77
78          Hitam membentuk d5 segera setelah c6, berusaha
79  menetralisir pusat putih tanpa menciptakan kelemahan
80  signifikan.
81      """.trimIndent(),
82                  imageResId = R.drawable.p4,
83                  link                                    =
84  "https://www.chess.com/openings/Caro-Kann-Defense"
85              ),
86
87              TampilanPembukaan(
88                  nama_pembukaan = "Italian Game",
89                  penjelasan_singkat = "Sebuah pembukaan
90  yang sering mengarah pada permainan terbuka, dimulai
91  dengan e4 e5 2.Nf3 Nc6 3.Bc4, dengan tujuan menyerang
92  pusat lawan.",
93                  penjelasan_dua_paragraf = """
94          Italian Game memberikan peluang pengembangan
95  cepat bagi kedua pihak. Putih langsung mengincar titik
96  lemah f7, titik lemah paling rentan bagi hitam di awal
97  permainan.
98
99          Pembukaan ini cocok untuk pemain pemula hingga
100 master karena keseimbangan antara taktik dan strategi.
101     """.trimIndent(),
102                 imageResId = R.drawable.p5,
103                 link                                    =
104 "https://www.chess.com/openings/Italian-Game"
105             ),
106
107             TampilanPembukaan(
108                 nama_pembukaan = "Queen's Gambit",
109                 penjelasan_singkat = "Pembukaan populer
110 yang dimulai dengan d4 d5 2.c4, di mana putih menawarkan
111 pion untuk mengontrol pusat papan.",
112                 penjelasan_dua_paragraf = """
113         Queen's Gambit adalah salah satu pembukaan
114 tertua dan paling dihormati dalam catur. Meski disebut
115 'gambit', pion yang dikorbankan biasanya dapat direbut
116 kembali.
117
118         Tujuan utama putih adalah mengalihkan pion d5
119 hitam dan menciptakan dominasi penuh di pusat papan.
120 Banyak juara dunia telah menggunakan pembukaan ini
121 dengan sukses besar.
122     """.trimIndent(),
123                 imageResId = R.drawable.p6,
124                 link                                    =
125 "https://www.chess.com/openings/Queens-Gambit"
```

```
            ),

            TampilanPembukaan(
                nama_pembukaan = "Slav Defense",
                penjelasan_singkat = "Dimulai dengan d4
d5 2.c4 c6, hitam bertujuan untuk menjaga pusat dan
bersiap untuk melawan serangan putih.",
                penjelasan_dua_paragraf = """
        Slav Defense adalah respon solid terhadap
Queen’s Gambit. Dengan memainkan c6, hitam memperkuat
kontrol atas d5 tanpa membuka terlalu banyak ruang.

        Ini menghasilkan posisi yang seimbang namun
fleksibel, memungkinkan transisi ke berbagai rencana
strategis tergantung perkembangan permainan.
    """.trimIndent(),
                imageResId = R.drawable.p7,
                link                                    =
"https://www.chess.com/openings/Slav-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan    =    "King's    Indian
Defense",
                penjelasan_singkat  =  "Strategi  yang
dimulai dengan 1.d4 Nf6 2.c4 g6, hitam berencana
menyerang pusat dengan pion dan pasukan yang
dikembangkan setelahnya.",
                penjelasan_dua_paragraf = """
        King's   Indian   Defense   dikenal   dengan
pendekatannya yang agresif dan asimetris. Hitam
mengizinkan putih membangun pusat besar, lalu
menyerangnya.

        Strategi khasnya adalah serangan raja oleh
hitam, bahkan ketika putih mengembangkan keunggulan
ruang. Ini adalah pilihan ideal bagi pecatur taktis.
    """.trimIndent(),
                imageResId = R.drawable.p8,
                link                                    =
"https://www.chess.com/openings/Kings-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan     =     "Nimzo-Indian
Defense",
                penjelasan_singkat = "Dimulai dengan d4
Nf6  2.c4  e6  3.Nc3  Bb4,  hitam  bertujuan  untuk

```

```
mengendalikan pusat sambil mengembangkan tekanan terhadap pion putih.",
                penjelasan_dua_paragraf = """
        Nimzo-Indian Defense adalah pembukaan strategis yang memanfaatkan ancaman terhadap struktur pion putih di awal. Dengan Bb4, hitam menekan Nc3 dan menciptakan potensi kerusakan struktur.

        Pembukaan ini menggabungkan kontrol pusat dengan tekanan posisi dan cocok bagi pemain yang menyukai fleksibilitas.
    """.trimIndent(),
                imageResId = R.drawable.p9,
                link = "https://www.chess.com/openings/Nimzo-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Queen's Indian Defense",
                penjelasan_singkat = "Dimulai dengan 1.d4 Nf6 2.c4 e6 3.Nf3 b6, hitam berusaha mengembangkan bidaknya dengan cara yang fleksibel dan mengontrol pusat.",
                penjelasan_dua_paragraf = """
        Queen’s Indian Defense fokus pada perkembangan bidak yang fleksibel dan pengendalian diagonal panjang dengan gajah di b7. Ini adalah pembukaan yang solid dan penuh manuver.

        Pembukaan ini cocok untuk pemain yang ingin menghindari konflik langsung di awal, namun siap melawan balik saat permainan berkembang.
    """.trimIndent(),
                imageResId = R.drawable.p10,
                link = "https://www.chess.com/openings/Queens-Indian-Defense"
            )

        )
    }
}
```

3. MainActivity.kt

Tabel 7 Source Code MainActivity.kt Modul 3

```
1   package com.example.pembukaan_catur
```

```
2
3   import android.content.Intent
4   import android.net.Uri
5   import android.os.Bundle
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.enableEdgeToEdge
9   import androidx.compose.foundation.Image
10  import androidx.compose.foundation.background
11  import androidx.compose.foundation.layout.Arrangement
12  import androidx.compose.foundation.layout.Column
13  import
14  androidx.compose.foundation.layout.PaddingValues
15  import androidx.compose.foundation.layout.Row
16  import androidx.compose.foundation.layout.Spacer
17  import androidx.compose.foundation.layout.WindowInsets
18  import
19  androidx.compose.foundation.layout.asPaddingValues
20  import
21  androidx.compose.foundation.layout.defaultMinSize
22  import androidx.compose.foundation.layout.fillMaxSize
23  import androidx.compose.foundation.layout.fillMaxWidth
24  import androidx.compose.foundation.layout.height
25  import androidx.compose.foundation.layout.padding
26  import androidx.compose.foundation.layout.size
27  import androidx.compose.foundation.layout.statusBars
28  import androidx.compose.foundation.layout.width
29  import
30  androidx.compose.foundation.layout.wrapContentWidth
31  import androidx.compose.foundation.lazy.LazyColumn
32  import androidx.compose.foundation.lazy.items
33  import
34  androidx.compose.foundation.shape.RoundedCornerShape
35  import androidx.compose.material3.Button
36  import androidx.compose.material3.ButtonDefaults
37  import androidx.compose.material3.Card
38  import androidx.compose.material3.CardDefaults
39  import androidx.compose.material3.MaterialTheme
40  import androidx.compose.material3.Scaffold
41  import androidx.compose.material3.Surface
42  import androidx.compose.material3.Text
43  import androidx.compose.runtime.Composable
44  import androidx.compose.ui.Alignment
45  import androidx.compose.ui.Modifier
46  import androidx.compose.ui.graphics.Color
47  import
48  androidx.compose.ui.layout.ModifierLocalBeyondBoundsLa
49  yout
50  import androidx.compose.ui.platform.LocalContext
```

```
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.tooling.preview.Preview
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import
com.example.pembukaan_catur.ui.theme.Pembukaan_caturTh
eme
import androidx.navigation.compose.NavHost
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.NavHost
import androidx.navigation.compose.composable
import
androidx.navigation.compose.rememberNavController
import androidx.navigation.navArgument
import kotlin.math.round

class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?)
{
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            Pembukaan_caturTheme {
                Surface(
                    modifier = Modifier.fillMaxSize(),
                ) {
                    val         navController        =
rememberNavController()
                    NavHost(
                        navController = navController,
                        startDestination             =
"PembukaanRepository"
                    ) {

composable("PembukaanRepository") {

DaftarPembukaan(navController)
                        }

                        composable(
                            route                    =
"penjelasan/{desc}/{img}",
                            arguments = listOf(
                                navArgument("desc")  {
type = NavType.StringType },
```

```
                                navArgument("img")    {
type = NavType.IntType }
                            )
                        ) { backStackEntry ->
                            val         desc          =
backStackEntry.arguments?.getString("desc") ?: ""
                            val         img           =
backStackEntry.arguments?.getInt("img") ?: 0
                            tampilanDetail(img  =  img,
nama = "Detail Pembukaan", penjelasan = desc)
                        }
                    }
                }
            }
        }
    }
}


// tampilan per entitas
@Composable
fun TampilanEntitasPembukaan(name: String, image: Int,
url: String, description: String, penjelasan: String,
navController: NavController) {
    val context = LocalContext.current
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(10.dp),
        shape = RoundedCornerShape(16.dp),
        elevation                                       =
CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Row(
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth(),
            verticalAlignment                           =
Alignment.CenterVertically
        ) {
            Image(
                painter = painterResource(id = image),
                contentDescription = null,
                modifier = Modifier
                    .size(width  =  120.dp,  height  =
120.dp)
            )
            Spacer(modifier = Modifier.width(16.dp))
```

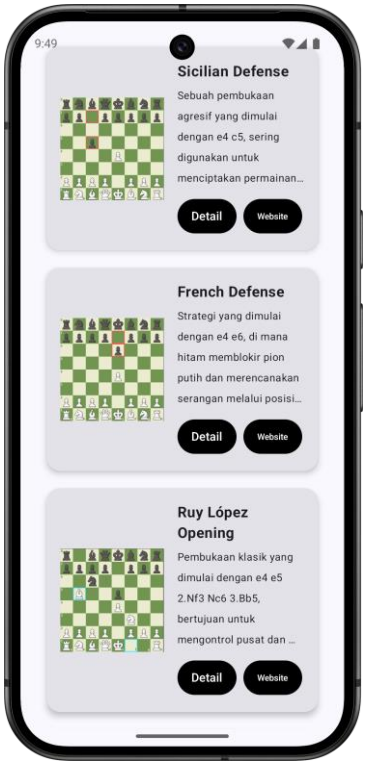
```
149             Column(
150                 modifier = Modifier
151                     .weight(1f)
152             ) {
153                 Text(text   =   name,   fontWeight   =
154 FontWeight.Bold, fontSize = 17.sp)
155                 Spacer(modifier                        =
156 Modifier.height(4.dp))
157                 Text(
158                     text = description,
159                     fontSize = 12.sp,
160                     maxLines = 5,
161                     overflow = TextOverflow.Ellipsis
162                 )
163                 Spacer(modifier                        =
164 Modifier.height(8.dp))
165                 Row(
166                     horizontalArrangement              =
167 Arrangement.SpaceBetween,
168                     modifier = Modifier
169                         .fillMaxWidth()
170
171 .wrapContentWidth(Alignment.Start),
172                 ) {
173                     Button(
174                         onClick = {
175                             val      encodedDesc       =
176 Uri.encode(penjelasan)
177
178 navController.navigate("penjelasan/$encodedDesc/$image
179 ")
180                         },
181                         contentPadding                 =
182 PaddingValues(horizontal = 16.dp, vertical = 8.dp),
183                         shape                          =
184 RoundedCornerShape(50),
185                         modifier                       =
186 Modifier.defaultMinSize(minWidth = 1.dp),
187                         colors                         =
188 ButtonDefaults.buttonColors(
189                             containerColor             =
190 Color.Black,
191                             contentColor = Color.White
192                         )
193                     ) {
194                         Text("Detail",    fontSize    =
195 14.sp, )
196                     }
197
```

```
198                     Spacer(modifier                     =
199 Modifier.width(8.dp))
200
201                     Button(
202                         onClick = {
203                             val        intent        =
204 Intent(Intent.ACTION_VIEW, Uri.parse(url))
205
206 context.startActivity(intent)
207                         },
208                         contentPadding                =
209 PaddingValues(horizontal = 16.dp, vertical = 8.dp),
210                         shape                         =
211 RoundedCornerShape(50),
212                         modifier                      =
213 Modifier.defaultMinSize(minWidth = 1.dp),
214                         colors                        =
215 ButtonDefaults.buttonColors(
216                             containerColor            =
217 Color.Black,
218                             contentColor = Color.White
219                         )
220                     ) {
221                         Text("Website",   fontSize   =
222 13.sp)
223                     }
224                 }
225             }
226         }
227     }
228 }
229
230 // tampilan looping untuk lazycolumn
231 @Composable
232 fun  DaftarPembukaan(navController:  NavHostController)
233 {
234
235     val repositoriPembukaan = PembukaanRepository()
236     val               fetchedPembukaan                =
237 repositoriPembukaan.ambilSemuaPembukaan()
238     LazyColumn(
239         modifier = Modifier
240             .fillMaxWidth()
241             .padding(20.dp)
242     ) {
243         items(items=fetchedPembukaan) {
244             pembukaan -> TampilanEntitasPembukaan(
245                 name = pembukaan.nama_pembukaan,
246                 image = pembukaan.imageResId,
```

```
                url = pembukaan.link,
                description                             =
pembukaan.penjelasan_singkat,
                penjelasan                              =
pembukaan.penjelasan_dua_paragraf,
                navController = navController
            )
        }
    }
}

@Composable
fun tampilanDetail(img: Int, nama: String, penjelasan:
String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)

.padding(WindowInsets.statusBars.asPaddingValues()),
        horizontalAlignment                             =
Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = img),
            contentDescription = "Gambar Detail dari
$nama",
            modifier = Modifier
                .fillMaxWidth()
                .height(380.dp)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = nama,
            fontSize = 30.sp,
            fontWeight = FontWeight.W800
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = penjelasan,
            fontSize = 20.sp,
        )
    }
}



@Preview(showBackground = true)
@Composable
```

```
296 fun GreetingPreview() {
297     Pembukaan_caturTheme {
298         tampilanDetail(R.drawable.p1,          "Sisilia
299 Defense", """
300         Sicilian Defense adalah salah satu pembukaan
301 paling populer dan tajam dalam catur modern. Dengan
302 membalas e4 dengan c5, hitam menghindari simetri dan
303 berusaha mengambil inisiatif dengan permainan sayap.
304
305         Pembukaan ini terkenal menghasilkan posisi yang
306 kompleks dan penuh taktik. Banyak juara dunia, seperti
307 Kasparov dan Fischer, mengandalkan Sicilian untuk
308 menghadapi pemain e4.
309     """.trimIndent())
310     }
311 }
```

B. Output Program



Gambar 13 Tampilan Awal UI Aplikasi Modul 3

Gambar 14 Tampilan Halaman Detail Modul 3

Gambar 15 Tampilan Saat Menuju Website Modul 3

C. Pembahasan

1. Pembukaan.kt

Pada file ini kita sebenarnya hanya membuat sebuah class dengan nama TampilanPembukaan dengan beberapa val parameter penting seperti nama_pembukaan dengan tipe data string, penjelasan_singkat dengan tipe data string, penjelasan_dua_paragraf dengan tipe data string, val dengan tipe data Drawable yang memiliki default value Int, dan yang terakhir adalah link dengan tipe data String. File ini akan digunakan data awal yang akan ditampilkan di bagian looping sesuai dengan MainActivity.kt nantinya.

2. PembukaanRepository.kt

Pada file ini kita membuat class dengan nama ambilSemuaPembukaan yang mengambil fungsi TampilanPembukaan dari file Pembukaan.kt dan akan mengembalikan fungsi itu dengan data-data yang dideklarasikan di file ini di dalam MainAcivity.kt

3. MainActivity.kt

Pada file MainActivity.kt ini seluruh komponen utama dari aplikasi kita didefinisikan dan dideklarasikan. Mulai dari tampilan daftar pembukaan, komponen per item, hingga tampilan detail. Pertama-tama, terdapat kelas MainActivity yang akan dieksekusi saat aplikasi pertama kali dijalankan. Pada fungsi onCreate, kita memanggil fungsi enableEdgeToEdge() untuk mengaktifkan tampilan layar secara penuh. Selanjutnya, komponen setContent digunakan untuk membungkus seluruh isi tampilan dengan tema Pembukaan_caturTheme. Kita juga memanggil Surface sebagai pembungkus utama dan NavHost digunakan untuk menangani navigasi antar halaman, dengan rute awal "PembukaanRepository".

Navigasi terdiri dari dua rute utama yaitu "PembukaanRepository" untuk menampilkan daftar pembukaan dan "penjelasan/{desc}/{img}" untuk menampilkan detail dari masing-masing pembukaan. Data seperti deskripsi dan gambar dikirim

melalui argumen menggunakan fungsi Uri.encode() agar dapat terbaca dengan benar di URL. Saat rute "penjelasan/{desc}/{img}" dijalankan, data akan diteruskan ke fungsi tampilanDetail() yang akan menampilkan gambar, nama, dan penjelasan lengkap yang berisi variabel penjelasan_dua_paragraf dari masing-masing pembukaan catur yang dipilih.

Komponen DaftarPembukaan berfungsi untuk menampilkan daftar seluruh pembukaan catur. Pada DaftarPembukaan, kita memanggil repository PembukaanRepository yang menyediakan data pembukaan. Kita menggunakan komponen LazyColumn untuk menampilkan data secara scrollable. Tiap item di dalamnya akan diteruskan ke fungsi TampilanEntitasPembukaan, yang bertugas untuk menampilkan satu entitas pembukaan catur lengkap dengan nama, gambar, penjelasan singkat, dan dua tombol sesuai dengan tampilan yang kita buat di fungsi TampilanEntitasPembukaan.

Fungsi TampilanEntitasPembukaan menampilkan komponen berbentuk Card, yang berisi gambar pembukaan di sebelah kiri dan informasi teks di sebelah kanan. Terdapat dua tombol yaitu “Detail” untuk menavigasi ke halaman penjelasan dan tombol “Website” yang membuka link ke sumber referensi yaitu chess.com menggunakan Intent.

Terakhir, fungsi tampilanDetail akan digunakan untuk menampilkan penjelasan lebih lengkap terkait satu pembukaan catur. Gambar akan ditampilkan di bagian atas, dilanjutkan dengan nama dan penjelasan dalam dua paragraf.

2. RecyclerView masih digunakan karena memberikan kontrol penuh atas perilaku dan performa daftar seperti pengelolaan tampilan yang banyak serta efisiensi memori yang baik, hal ini menjadi sangat penting untuk aplikasi kita apabila aplikasi kita sudah berskala besar atau dengan data dinamis. Meskipun kode RecyclerView cenderung boiler-plate, yang berarti fleksibilitasnya lebih tinggi dibandingkan *LazyColumn* di Jetpack Compose, hal ini menyebabkan bahwa RecyclerView lebih cocok untuk antarmuka deklaratif dan kebutuhan yang lebih sederhana atau modern.

Pada bagian awal halaman HTML, terdapat sebuah <form> yang digunakan untuk meminta input dari user untuk nilai jumlah peserta. Ketika tombol "Cetak" diklik, data akan dikirim menggunakan metode POST. Setelah tombol diklik, bagian PHP akan dijalankan. Pertama-tama, PHP akan memeriksa apakah tombol "Cetak" telah ditekan dengan isset($_POST['cetak']). Jika iya, maka data dari input jumlah peserta akan disimpan ke variabel $banyak. Selanjutnya akan dilakukan perulangan menggunakan while mulai dari 1 sampai nilai jumlah peserta yang dimasukkan.

Di dalam perulangan, setiap angka peserta dicek apakah ganjil atau genap. Jika angka ganjil, maka akan dicetak dalam div kelas ganjil (berwarna merah) dan jika genap akan dicetak dalam div kelas genap (berwarna hijau). Hasilnya adalah daftar peserta dengan nomor ganjil berwarna merah dan nomor genap berwarna hijau, yang ditampilkan dalam format heading <h2>.

MODUL 4: ViewModel and Debugging

SOAL 1

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

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

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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

A. Source Code

1. Pembukaan.kt

Tabel 8 Source Code Pembukaan.kt Modul 4

```
1  package com.example.pembukaan_catur
2
3  import androidx.annotation.DrawableRes
```

```
4
5   data class TampilanPembukaan(
6       val nama_pembukaan: String,
7       val penjelasan_singkat: String,
8       val penjelasan_dua_paragraf: String,
9       @DrawableRes val imageResId: Int,
10      val link: String
11  )
```

2. PembukaanRepository.kt

Tabel 9 Source Code PembukaanRepository.kt Modul 4

```
1   package com.example.pembukaan_catur
2
3   class PembukaanRepository {
4       fun ambilSemuaPembukaan(): List<TampilanPembukaan>{
5           return listOf(
6               TampilanPembukaan(
7                   nama_pembukaan = "Sicilian Defense",
8                   penjelasan_singkat = "Sebuah pembukaan
9   agresif yang dimulai dengan e4 c5, sering digunakan untuk
10  menciptakan permainan tidak seimbang.",
11                  penjelasan_dua_paragraf = """
12          Sicilian Defense adalah salah satu pembukaan
13  paling populer dan tajam dalam catur modern. Dengan
14  membalas e4 dengan c5, hitam menghindari simetri dan
15  berusaha mengambil inisiatif dengan permainan sayap.
16
17          Pembukaan ini terkenal menghasilkan posisi yang
18  kompleks dan penuh taktik. Banyak juara dunia, seperti
19  Kasparov dan Fischer, mengandalkan Sicilian untuk
20  menghadapi pemain e4.
21      """.trimIndent(),
22                  imageResId = R.drawable.p1,
23                  link                                    =
24  "https://www.chess.com/openings/Sicilian-Defense"
25              ),
26
27              TampilanPembukaan(
28                  nama_pembukaan = "French Defense",
29                  penjelasan_singkat  =  "Strategi  yang
30  dimulai dengan e4 e6, di mana hitam memblokir pion putih
31  dan merencanakan serangan melalui posisi yang lebih
32  tertutup.",
33                  penjelasan_dua_paragraf = """
34          French Defense memberikan struktur pion yang
35  kokoh dan solid untuk hitam. Ia menawarkan banyak
36  kemungkinan strategis, terutama dalam permainan tengah.
```

```
37
38          Salah satu ciri khasnya adalah konflik antara
39  pion e5 putih dan struktur pertahanan hitam di d5. Taktik
40  dan strategi memainkan peran penting dalam membuka posisi
41  ini.
42      """.trimIndent(),
43                  imageResId = R.drawable.p2,
44                  link                                    =
45  "https://www.chess.com/openings/French-Defense"
46              ),
47
48              TampilanPembukaan(
49                  nama_pembukaan = "Ruy López Opening",
50                  penjelasan_singkat = "Pembukaan klasik
51  yang dimulai dengan e4 e5 2.Nf3 Nc6 3.Bb5, bertujuan untuk
52  mengontrol pusat dan menekan pertahanan hitam.",
53                  penjelasan_dua_paragraf = """
54          Ruy López adalah salah satu pembukaan tertua yang
55  masih dimainkan di level tinggi. Dengan menekan kuda di
56  c6, putih mencoba mengganggu kontrol hitam terhadap
57  pusat.
58
59          Posisi yang timbul sering bersifat strategis,
60  dengan ruang untuk manuver jangka panjang dan potensi
61  serangan raja di tahap akhir pembukaan.
62      """.trimIndent(),
63                  imageResId = R.drawable.p3,
64                  link                                    =
65  "https://www.chess.com/openings/Ruy-Lopez-Opening"
66              ),
67
68              TampilanPembukaan(
69                  nama_pembukaan = "Caro-Kann Defense",
70                  penjelasan_singkat = "Dimulai dengan e4
71  c6, hitam berusaha untuk memperkuat pusat dan membangun
72  pertahanan yang solid.",
73                  penjelasan_dua_paragraf = """
74          Caro-Kann terkenal karena stabilitas dan keamanan
75  bagi raja hitam. Ini adalah pembukaan pilihan bagi pemain
76  yang menyukai posisi bertahan namun aktif.
77
78          Hitam membentuk d5 segera setelah c6, berusaha
79  menetralisir pusat putih tanpa menciptakan kelemahan
80  signifikan.
81      """.trimIndent(),
82                  imageResId = R.drawable.p4,
83                  link                                    =
84  "https://www.chess.com/openings/Caro-Kann-Defense"
85              ),
```

```
86
87              TampilanPembukaan(
88                  nama_pembukaan = "Italian Game",
89                  penjelasan_singkat = "Sebuah pembukaan
90  yang sering mengarah pada permainan terbuka, dimulai
91  dengan e4 e5 2.Nf3 Nc6 3.Bc4, dengan tujuan menyerang
92  pusat lawan.",
93                  penjelasan_dua_paragraf = """
94          Italian Game memberikan peluang pengembangan
95  cepat bagi kedua pihak. Putih langsung mengincar titik
96  lemah f7, titik lemah paling rentan bagi hitam di awal
97  permainan.
98
99          Pembukaan ini cocok untuk pemain pemula hingga
100 master karena keseimbangan antara taktik dan strategi.
101     """.trimIndent(),
102                 imageResId = R.drawable.p5,
103                 link                                    =
104 "https://www.chess.com/openings/Italian-Game"
105             ),
106
107             TampilanPembukaan(
108                 nama_pembukaan = "Queen's Gambit",
109                 penjelasan_singkat = "Pembukaan populer
110 yang dimulai dengan d4 d5 2.c4, di mana putih menawarkan
111 pion untuk mengontrol pusat papan.",
112                 penjelasan_dua_paragraf = """
113         Queen's Gambit adalah salah satu pembukaan tertua
114 dan paling dihormati dalam catur. Meski disebut 'gambit',
115 pion yang dikorbankan biasanya dapat direbut kembali.
116
117         Tujuan utama putih adalah mengalihkan pion d5
118 hitam dan menciptakan dominasi penuh di pusat papan.
119 Banyak juara dunia telah menggunakan pembukaan ini dengan
120 sukses besar.
121     """.trimIndent(),
122                 imageResId = R.drawable.p6,
123                 link                                    =
124 "https://www.chess.com/openings/Queens-Gambit"
125             ),
126
127             TampilanPembukaan(
128                 nama_pembukaan = "Slav Defense",
129                 penjelasan_singkat = "Dimulai dengan d4
130 d5 2.c4 c6, hitam bertujuan untuk menjaga pusat dan
131 bersiap untuk melawan serangan putih.",
132                 penjelasan_dua_paragraf = """
133
134
```

```
        Slav Defense adalah respon solid terhadap Queen's Gambit. Dengan memainkan c6, hitam memperkuat kontrol atas d5 tanpa membuka terlalu banyak ruang.

        Ini menghasilkan posisi yang seimbang namun fleksibel, memungkinkan transisi ke berbagai rencana strategis tergantung perkembangan permainan.
    """.trimIndent(),
                imageResId = R.drawable.p7,
                link                                    = "https://www.chess.com/openings/Slav-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan    =    "King's    Indian Defense",
                penjelasan_singkat  =  "Strategi  yang dimulai dengan 1.d4 Nf6 2.c4 g6, hitam berencana menyerang pusat dengan pion dan pasukan yang dikembangkan setelahnya.",
                penjelasan_dua_paragraf = """
        King's    Indian    Defense    dikenal    dengan pendekatannya yang agresif dan asimetris. Hitam mengizinkan putih membangun pusat besar, lalu menyerangnya.

        Strategi khasnya adalah serangan raja oleh hitam, bahkan ketika putih mengembangkan keunggulan ruang. Ini adalah pilihan ideal bagi pecatur taktis.
    """.trimIndent(),
                imageResId = R.drawable.p8,
                link                                    = "https://www.chess.com/openings/Kings-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan = "Nimzo-Indian Defense",
                penjelasan_singkat = "Dimulai dengan d4 Nf6  2.c4  e6  3.Nc3  Bb4,  hitam  bertujuan  untuk mengendalikan pusat sambil mengembangkan tekanan terhadap pion putih.",
                penjelasan_dua_paragraf = """
        Nimzo-Indian Defense adalah pembukaan strategis yang memanfaatkan ancaman terhadap struktur pion putih di awal. Dengan Bb4, hitam menekan Nc3 dan menciptakan potensi kerusakan struktur.
```

```
        Pembukaan ini menggabungkan kontrol pusat dengan
tekanan posisi dan cocok bagi pemain yang menyukai
fleksibilitas.
    """.trimIndent(),
                imageResId = R.drawable.p9,
                link                                        =
"https://www.chess.com/openings/Nimzo-Indian-Defense"
            ),

            TampilanPembukaan(
                nama_pembukaan    =    "Queen's    Indian
Defense",
                penjelasan_singkat  =  "Dimulai  dengan
1.d4 Nf6 2.c4 e6 3.Nf3 b6, hitam berusaha mengembangkan
bidaknya dengan cara yang fleksibel dan mengontrol
pusat.",
                penjelasan_dua_paragraf = """
        Queen’s Indian Defense fokus pada perkembangan
bidak yang fleksibel dan pengendalian diagonal panjang
dengan gajah di b7. Ini adalah pembukaan yang solid dan
penuh manuver.

        Pembukaan ini cocok untuk pemain yang ingin
menghindari konflik langsung di awal, namun siap melawan
balik saat permainan berkembang.
    """.trimIndent(),
                imageResId = R.drawable.p10,
                link                                        =
"https://www.chess.com/openings/Queens-Indian-Defense"
            )

        )
    }
}
```

3. PembukaanViewModel.kt

Tabel 10 Source Code viewmodel/PembukaanViewModel.kt Modul 4

```
package com.example.pembukaan_catur.viewmodel

import androidx.lifecycle.ViewModel
import
com.example.pembukaan_catur.model.TampilanPembukaan
import
com.example.pembukaan_catur.model.PembukaanRepository
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import android.util.Log

```

```
class PembukaanViewModel(private val repository:
PembukaanRepository) : ViewModel() {

    private val _pembukaanList =
MutableStateFlow<List<TampilanPembukaan>>(emptyList())
    val pembukaanList:
StateFlow<List<TampilanPembukaan>> = _pembukaanList

    private val _selectedPembukaan =
MutableStateFlow<TampilanPembukaan?>(null)
    val selectedPembukaan:
StateFlow<TampilanPembukaan?> = _selectedPembukaan

    init {
        val data = repository.ambilSemuaPembukaan()
        _pembukaanList.value = data

        Log.d("PembukaanViewModel", "Data item dimuat
ke dalam list: ${data.size} items")
    }

    fun pilihPembukaan(pembukaan: TampilanPembukaan) {
        _selectedPembukaan.value = pembukaan
        Log.d("PembukaanViewModel", "Item dipilih untuk
detail: ${pembukaan.nama_pembukaan}")
    }

    fun logTombolDetail(item: TampilanPembukaan) {
        Log.d("PembukaanViewModel", "Tombol Detail
ditekan: ${item.nama_pembukaan}")
    }

    fun logTombolExplicitIntent(item:
TampilanPembukaan) {
        Log.d("PembukaanViewModel", "Tombol Website
ditekan: ${item.nama_pembukaan}")
    }
}
```

4. PembukaanviewModelFactory.kt

Tabel 11 Source Code viewmodel/PembukaanviewModelFactory.kt Modul 4

```
package com.example.pembukaan_catur.viewmodel

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

```
com.example.pembukaan_catur.model.PembukaanRepository

class PembukaanViewModelFactory(
    private val repository: PembukaanRepository
) : ViewModelProvider.Factory {

    @Suppress("UNCHECKED_CAST")
    override fun <T : ViewModel> create(modelClass:
Class<T>): T {
        if
(modelClass.isAssignableFrom(PembukaanViewModel::class.java)) {
            return PembukaanViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel
class")
    }
}
```

5. MainActivity.kt

Tabel 12 Source Code MainActivity.kt Modul 4

```
package com.example.pembukaan_catur
import android.content.Intent
import android.net.Uri
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.enableEdgeToEdge
import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
```

```
import
androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.ViewModelProvider
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import androidx.navigation.NavHostController
import androidx.navigation.NavType
import androidx.navigation.compose.*
import androidx.navigation.navArgument
import
com.example.pembukaan_catur.model.PembukaanRepository
import
com.example.pembukaan_catur.ui.theme.Pembukaan_caturThe
me
import
com.example.pembukaan_catur.viewmodel.PembukaanViewMode
l
import
com.example.pembukaan_catur.viewmodel.PembukaanViewMode
lFactory
import kotlinx.coroutines.flow.collectLatest

class MainActivity : ComponentActivity() {

    private lateinit var pembukaanViewModelFactory:
PembukaanViewModelFactory
    private lateinit var pembukaanViewModel:
PembukaanViewModel

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()

        pembukaanViewModelFactory =
PembukaanViewModelFactory(PembukaanRepository())
        pembukaanViewModel = ViewModelProvider(this,
pembukaanViewModelFactory)[PembukaanViewModel::class.ja
va]
```

```
131 
132 
133         setContent {
134             Pembukaan_caturTheme {
135                 Surface(modifier                        =
136 Modifier.fillMaxSize()) {
137                     val          navController          =
138 rememberNavController()
139                     NavHost(
140                         navController = navController,
141                         startDestination                =
142 "listPembukaan"
143                     ) {
144                         composable("listPembukaan") {
145                             //  Ambil  state  list  dari
146 ViewModel dengan collectAsState()
147                             val     pembukaanList     by
148 pembukaanViewModel.pembukaanList.collectAsState()
149                             val         context         =
150 LocalContext.current
151 
152                             DaftarPembukaan(
153                                 pembukaanItems          =
154 pembukaanList,
155                                 navController          =
156 navController,
157                                 onDetailClick = { item -
158 >
159 
160 pembukaanViewModel.logTombolDetail(item)     //   Panggil
161 tanpa argumen
162                                     val   encodedDesc   =
163 Uri.encode(item.penjelasan_dua_paragraf)
164 
165 navController.navigate("penjelasan/$encodedDesc/${item.
166 imageResId}")
167                                 },
168 
169                                 onWebsiteClick = { item
170 ->
171 
172 pembukaanViewModel.logTombolExplicitIntent(item)
173                                     val     intent     =
174 Intent(Intent.ACTION_VIEW, Uri.parse(item.link))
175 
176 context.startActivity(intent)
177                                 }
178                             )
179 
```

```
                        }
                        composable(
                            route                       =
"penjelasan/{desc}/{img}",
                            arguments = listOf(
                                navArgument("desc")    {
type = NavType.StringType },
                                navArgument("img")     {
type = NavType.IntType }
                            )
                        ) { backStackEntry ->
                            val          desc           =
backStackEntry.arguments?.getString("desc") ?: ""
                            val           img           =
backStackEntry.arguments?.getInt("img") ?: 0

                            //  Log  saat  berpindah  ke
halaman Detail
                            Log.d("PembukaanViewModel",
"Navigasi ke detail dengan deskripsi: $desc")

                            tampilanDetail(img  =  img,
nama = "Detail Pembukaan", penjelasan = desc)
                        }
                    }
                }
            }
        }
    }
}

// Composable daftar pembukaan dengan callback klik
@Composable
fun DaftarPembukaan(
    pembukaanItems:
List<com.example.pembukaan_catur.model.TampilanPembukaa
n>,
    navController: NavHostController,
    onDetailClick:
(com.example.pembukaan_catur.model.TampilanPembukaan) ->
Unit,
    onWebsiteClick:
(com.example.pembukaan_catur.model.TampilanPembukaan) ->
Unit
) {
    LazyColumn(
        modifier = Modifier
            .fillMaxWidth()
            .padding(20.dp)
```

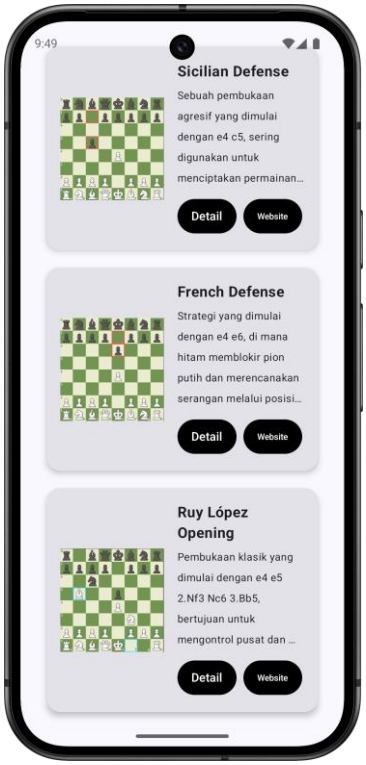
```
    ) {
        items(items = pembukaanItems) { item ->
            TampilanEntitasPembukaan(
                name = item.nama_pembukaan,
                image = item.imageResId,
                url = item.link,
                description = item.penjelasan_singkat,
                penjelasan                                    =
item.penjelasan_dua_paragraf,
                navController = navController,
                onDetailClick = { onDetailClick(item) },
                onWebsiteClick = { onWebsiteClick(item)
}
            )
        }
    }
}

// Perbaikan fungsi TampilanEntitasPembukaan dengan
callback tombol
@Composable
fun TampilanEntitasPembukaan(
    name: String,
    image: Int,
    url: String,
    description: String,
    penjelasan: String,
    navController: NavController,
    onDetailClick: () -> Unit,
    onWebsiteClick: () -> Unit
) {
    val context = LocalContext.current
    Card(
        modifier = Modifier
            .fillMaxWidth()
            .padding(10.dp),
        shape = RoundedCornerShape(16.dp),
        elevation                                             =
CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Row(
            modifier = Modifier
                .padding(16.dp)
                .fillMaxWidth(),
            verticalAlignment                                 =
Alignment.CenterVertically
        ) {
            Image(
                painter = painterResource(id = image),
```

```
278                 contentDescription = null,
279                 modifier = Modifier
280                     .size(width  =  120.dp,  height  =
281 120.dp)
282             )
283             Spacer(modifier = Modifier.width(16.dp))
284             Column(
285                 modifier = Modifier
286                     .weight(1f)
287             ) {
288                 Text(text   =   name,   fontWeight   =
289 FontWeight.Bold, fontSize = 17.sp)
290                 Spacer(modifier = Modifier.height(4.dp))
291                 Text(
292                     text = description,
293                     fontSize = 12.sp,
294                     maxLines = 5,
295                     overflow = TextOverflow.Ellipsis
296                 )
297                 Spacer(modifier = Modifier.height(8.dp))
298                 Row(
299                     horizontalArrangement              =
300 Arrangement.SpaceBetween,
301                     modifier = Modifier
302                         .fillMaxWidth()
303
304 .wrapContentWidth(Alignment.Start),
305                 ) {
306                     Button(
307                         onClick = onDetailClick,
308                         contentPadding                 =
309 PaddingValues(horizontal = 16.dp, vertical = 8.dp),
310                         shape = RoundedCornerShape(50),
311                         modifier                       =
312 Modifier.defaultMinSize(minWidth = 1.dp),
313                         colors                         =
314 ButtonDefaults.buttonColors(
315                             containerColor             =
316 Color.Black,
317                             contentColor = Color.White
318                         )
319                     ) {
320                         Text("Detail", fontSize = 14.sp)
321                     }
322
323                     Spacer(modifier                    =
324 Modifier.width(8.dp))
325
326                     Button(
```

```
                            onClick = onWebsiteClick,
                            contentPadding                  =
PaddingValues(horizontal = 16.dp, vertical = 8.dp),
                            shape = RoundedCornerShape(50),
                            modifier                        =
Modifier.defaultMinSize(minWidth = 1.dp),
                            colors                          =
ButtonDefaults.buttonColors(
                                containerColor              =
Color.Black,
                                contentColor = Color.White
                            )
                        ) {
                            Text("Website",    fontSize    =
10.sp)
                        }
                    }
                }
            }
        }
    }
}

@Composable
fun tampilanDetail(img: Int, nama: String, penjelasan:
String) {
    Column(
        modifier = Modifier
            .fillMaxSize()
            .padding(16.dp)

.padding(WindowInsets.statusBars.asPaddingValues()),
        horizontalAlignment                                 =
Alignment.CenterHorizontally
    ) {
        Image(
            painter = painterResource(id = img),
            contentDescription  =  "Gambar  Detail  dari
$nama",
            modifier = Modifier
                .fillMaxWidth()
                .height(380.dp)
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            text = nama,
            fontSize = 30.sp,
            fontWeight = FontWeight.W800
        )
        Spacer(modifier = Modifier.height(16.dp))
```

```
            Text(
                text = penjelasan,
                fontSize = 20.sp,
            )
        }
}
```

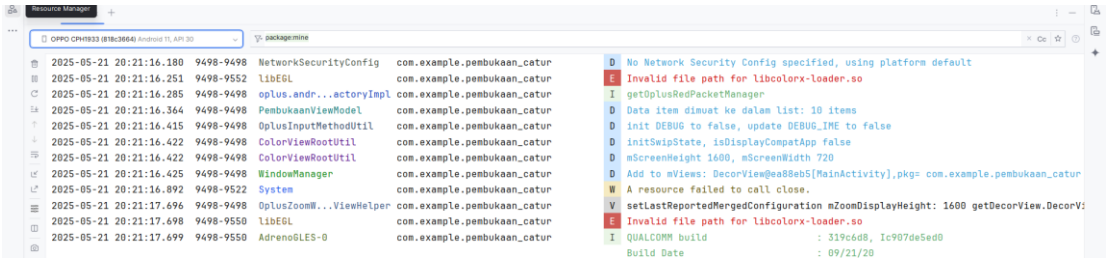
B. Output Program



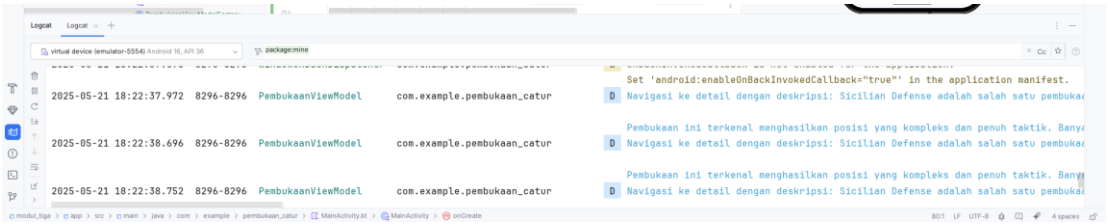
Gambar 16 Tampilan Awal UI Aplikasi Modul 4

Gambar 17 Tampilan Halaman Detail Modul 4

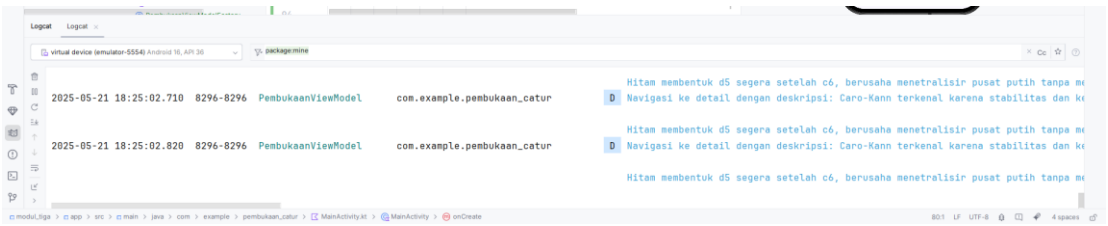
Gambar 18 Tampilan Saat Menuju Website Modul 4



Gambar 19 Log saat data item masuk ke dalam list Modul 4



Gambar 20 Log saat tombol Detail dan tombol Explicit Intent ditekan Modul 4



Gambar 21 Log data ketika berpindah ke halaman Detail Modul 4

C. Pembahasan

1. Pembukaan.kt

Pada file ini kita sebenarnya hanya membuat sebuah class dengan nama TampilanPembukaan dengan beberapa val parameter penting seperti nama_pembukaan dengan tipe data string, penjelasan_singkat dengan tipe data string, penjelasan_dua_paragraf dengan tipe data string, val dengan tipe data Drawable yang memiliki default value Int, dan yang terakhir adalah link dengan tipe data String. File ini akan digunakan data awal yang akan ditampilkan di bagian looping sesuai dengan MainActivity.kt nantinya.

2. PembukaanRepository.kt

Pada file ini kita membuat class dengan nama ambilSemuaPembukaan yang mengambil fungsi TampilanPembukaan dari file Pembukaan.kt dan akan mengembalikan fungsi itu dengan data-data yang dideklarasikan di file ini di dalam MainAcivity.kt

3. PembukaanViewModel.kt

Pada file ini, kita membuat sebuah kelas bernama PembukaanViewModel yang mewarisi dari ViewModel. File ini bertanggung jawab untuk mengatur dan mengelola data yang akan ditampilkan di tampilan UI, khususnya data yang berhubungan dengan pembukaan catur. Di dalam kelas ini, terdapat dua buah StateFlow, yaitu _pembukaanList yang berisi daftar seluruh pembukaan catur dan _selectedPembukaan yang menyimpan data dari satu item pembukaan yang dipilih oleh pengguna. Data pembukaan ini sendiri diambil dari PembukaanRepository, yang sebelumnya sudah di-inject lewat constructor.

Pada bagian init, semua data pembukaan langsung dimuat dari repository dan dimasukkan ke dalam _pembukaanList, sehingga nantinya bisa diamati oleh tampilan antarmuka. Selain itu, terdapat beberapa fungsi penting seperti pilihPembukaan() yang digunakan untuk menetapkan pembukaan yang sedang dipilih, serta dua fungsi logging tambahan yaitu logTombolDetail() dan logTombolExplicitIntent() yang hanya bertugas mencatat ke logcat saat tombol-tombol tertentu ditekan.

4. PembukaanViewModelFactory.kt

Pada file ini, kita membuat sebuah class dengan nama PembukaanViewModelFactory yang berfungsi sebagai factory atau pabrik pembuat objek PembukaanViewModel. Kelas ini mengimplementasikan interface ViewModelProvider.Factory, yang memang biasa digunakan ketika kita ingin mengirim parameter ke dalam sebuah ViewModel. Dalam kasus ini, parameter yang dikirim adalah repository, yaitu instance dari PembukaanRepository yang akan digunakan oleh ViewModel nantinya.

Di dalam fungsi create(), kita melakukan pengecekan terlebih dahulu apakah modelClass yang diminta merupakan turunan dari PembukaanViewModel. Jika ya,

maka kita akan mengembalikan instance dari PembukaanViewModel dengan repository sebagai parameternya. Namun jika tidak cocok, maka akan dilemparkan sebuah exception dengan pesan “Unknown ViewModel class”.

5. MainActivity.kt

File MainActivity.kt merupakan titik masuk utama dari aplikasi, yang berfungsi untuk mengatur tampilan dan navigasi antar layar menggunakan Jetpack Compose. Di dalam onCreate, kita menginisialisasi PembukaanViewModelFactory dengan menyisipkan PembukaanRepository sebagai dependensinya, lalu menggunakan factory ini untuk mendapatkan instance dari PembukaanViewModel melalui ViewModelProvider. Ini memungkinkan ViewModel digunakan secara terpusat dan dapat mengakses data dari repository.

Selanjutnya, fungsi setContent digunakan untuk menyusun UI menggunakan Composable dalam tema Pembukaan_caturTheme. Di dalamnya terdapat struktur navigasi NavHost dengan dua rute utama, yaitu "listPembukaan" dan "penjelasan/{desc}/{img}". Rute listPembukaan menampilkan daftar pembukaan catur yang diambil dari pembukaanList, yaitu StateFlow yang dikoleksi menggunakan collectAsState. Masing-masing entri daftar ditampilkan melalui komponen TampilanEntitasPembukaan, lengkap dengan tombol untuk melihat detail atau membuka link website eksternal, dengan intent eksplisit.

Jika pengguna menekan tombol “Detail”, maka aplikasi akan melakukan navigasi ke halaman penjelasan menggunakan rute "penjelasan/{desc}/{img}". Parameter desc dan img akan diterima pada halaman tujuan sebagai argumen. Halaman penjelasan tersebut ditampilkan melalui fungsi tampilanDetail, yang menunjukkan gambar pembukaan catur, nama, dan penjelasan lengkap. Seluruh struktur ini menunjukkan penggunaan prinsip arsitektur MVVM dan Compose Navigation secara efektif dan modular, serta memanfaatkan pendekatan reactive UI dari Jetpack Compose.

2. RecyclerView masih digunakan karena memberikan kontrol penuh atas perilaku dan performa daftar seperti pengelolaan tampilan yang banyak serta efisiensi memori yang baik, hal ini menjadi sangat penting untuk aplikasi kita apabila aplikasi kita sudah berskala besar atau dengan data dinamis. Meskipun kode RecyclerView cenderung boiler-plate, yang berarti fleksibilitasnya lebih tinggi dibandingkan *LazyColumn* di Jetpack Compose, hal ini menyebabkan bahwa RecyclerView lebih cocok untuk antarmuka deklaratif dan kebutuhan yang lebih sederhana atau modern.

MODUL 5: Connect to the Internet

SOAL 1

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
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. model/Product.kt

Tabel 13 Source Code model/Product.kt Modul 5

```
1  package com.example.modul_5.data.model
2
3  import kotlinx.serialization.Serializable
4
5  @Serializable
6  data class Product(
7      val id: Int,
8      val title: String,
```

```
    val price: Double,
    val description: String,
    val category: String,
    val image: String,
    val rating: Rating
)

@Serializable
data class Rating(
    val rate: Double,
    val count: Int
)
```

2. remote/ProductApiService.kt

Tabel 14 Source Code remote/ProductApiService.kt Modul 5

```
package com.example.modul_5.data.remote

import com.example.modul_5.data.model.Product
import retrofit2.http.GET

interface ProductApiService {
    @GET("products")
    suspend fun getProducts(): List<Product>
}
```

3. remote/RetrofitInstance.kt

Tabel 15 Source Code remote/RetrofitInstance.kt Modul 5

```
package com.example.modul_5.data.remote

import
com.jakewharton.retrofit2.converter.kotlinx.serializatio
n.asConverterFactory
import
kotlinx.serialization.ExperimentalSerializationApi
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import retrofit2.Retrofit

object RetrofitInstance {
    private val json = Json {
        ignoreUnknownKeys = true
    }

```

```
    @OptIn(ExperimentalSerializationApi::class)
    val api: ProductApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://fakestoreapi.com/")

.addConverterFactory(json.asConverterFactory("applicatio
n/json".toMediaType()))
            .build()
            .create(ProductApiService::class.java)
    }
}
```

4. repository/ProductRepository.kt

Tabel 16 Source Code repository/ProductRepository.kt Modul 5

```
package com.example.modul_5.repository

import android.util.Log
import com.example.modul_5.data.model.Product
import com.example.modul_5.data.remote.RetrofitInstance
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow

class ProductRepository {
    fun getProducts(): Flow<Result<List<Product>>> =
flow {
        try {
            val products =
RetrofitInstance.api.getProducts()
            Log.d("ProductRepository", "Success fetching
products: ${products.size} items")
            emit(Result.success(products))
        } catch (e: Exception) {
            Log.e("ProductRepository", "Error fetching
products", e)
            emit(Result.failure(e))
        }
    }
}
```

5. ui/detail/DetailScreen.kt

Tabel 17 Source Code ui/detail/DetailScreen.kt Modul 5

```
package com.example.modul_5.ui.detail

import androidx.compose.foundation.layout.*
import androidx.compose.material3.*
```

```
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.unit.dp
import com.example.modul_5.data.model.Product
import com.example.modul_5.ui.home.HomeViewModel



@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun DetailScreen(productId: Int, viewModel:
HomeViewModel) {
    val products by viewModel.products.collectAsState()
    val isLoading by
viewModel.isLoading.collectAsState()
    val error by viewModel.error.collectAsState()

    val product = products.find { it.id == productId }

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Detail Produk") }
            )
        }
    ) { paddingValues ->
        Box(
            modifier = Modifier
                .padding(paddingValues)
                .fillMaxSize(),
            contentAlignment = Alignment.Center
        ) {
            when {
                isLoading -> {
                    CircularProgressIndicator()
                }
                error != null -> {
                    Text(
                        text = error ?: "Terjadi
kesalahan",
                        color =
MaterialTheme.colorScheme.error
                    )
                }
                product != null -> {
                    ProductDetailContent(product =
product)
                }

```

```
                    else -> {
                        Text("Produk tidak ditemukan.")
                    }
                }
            }
        }
}

@Composable
fun ProductDetailContent(product: Product) {
    Column(
        modifier = Modifier
            .fillMaxWidth()
            .padding(16.dp),
        verticalArrangement = Arrangement.spacedBy(8.dp)
    ) {
        Text("Nama: ${product.title}", style =
MaterialTheme.typography.titleMedium)
        Text("Harga: $${product.price}", style =
MaterialTheme.typography.bodyMedium)
        Text("Kategori: ${product.category}", style =
MaterialTheme.typography.bodyMedium)
        Text("Deskripsi: ${product.description}", style
= MaterialTheme.typography.bodySmall)
    }
}
```

6. ui/home/HomeScreen.kt

Tabel 18 Source Code ui/home/HomeScreen.kt Modul 5

```
@file:OptIn(ExperimentalMaterial3Api::class)

package com.example.modul_5.ui.home
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.items
import androidx.compose.material3.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.navigation.NavController
import androidx.navigation.NavType
import androidx.navigation.compose.*
import androidx.navigation.navArgument
import coil.compose.AsyncImage
```

```
import com.example.modul_5.data.model.Product
import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.background
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext



@Composable
fun HomeNavHost(
    viewModel: HomeViewModel,
    modifier: Modifier = Modifier
) {
    val navController = rememberNavController()

    NavHost(
        navController = navController,
        startDestination = "home",
        modifier = modifier
    ) {
        composable("home") {
            HomeScreen(viewModel = viewModel,
navController = navController)
        }
        composable(
            route = "detail/{productId}",
            arguments = listOf(navArgument("productId")
{ type = NavType.IntType })
        ) { backStackEntry ->
            val productId =
backStackEntry.arguments?.getInt("productId") ?: 0
            DetailScreen(productId = productId,
viewModel = viewModel)
        }
    }
}



@Composable
fun HomeScreen(
    viewModel: HomeViewModel,
    navController: NavController,
) {

    val context = LocalContext.current

    val products by viewModel.products.collectAsState()
```

```
    val isLoading by
viewModel.isLoading.collectAsState()
    val error by viewModel.error.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Product List") }
            )
        }
    ) { paddingValues ->
        Box(
            modifier = Modifier
                .padding(paddingValues)
                .fillMaxSize()
        ) {
            when {
                isLoading -> {
                    CircularProgressIndicator(modifier
= Modifier.align(Alignment.Center))
                }
                error != null -> {
                    Text(
                        text = error ?: "Unknown
error",
                        modifier =
Modifier.align(Alignment.Center),
                        color =
MaterialTheme.colorScheme.error
                    )
                }
                else -> {
                    LazyColumn(
                        modifier =
Modifier.fillMaxSize(),
                        contentPadding =
PaddingValues(16.dp),
                        verticalArrangement =
Arrangement.spacedBy(12.dp)
                    ) {
                        items(products) { product ->
                            ProductCard(product =
product, onDetailClick = {

navController.navigate("detail/${product.id}")
                            }, onWebsiteClick = {
                                val searchQuery =
Uri.encode(product.title)

```

```
                                    val url =
"https://shopee.co.id/search?keyword=$searchQuery"
                                    val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(url))

context.startActivity(intent)
                                })
                            }
                        }
                    }
                }
            }
        }
}

@Composable
fun ProductCard(
    product: Product,
    onDetailClick: () -> Unit,
    onWebsiteClick: () -> Unit
) {
    val context = LocalContext.current

    Card(
        modifier = Modifier.fillMaxWidth(),
        elevation =
CardDefaults.cardElevation(defaultElevation = 6.dp),
        colors = CardDefaults.cardColors(containerColor
= Color.White)
    ) {
        Column(modifier = Modifier.padding(12.dp)) {
            AsyncImage(
                model = product.image,
                contentDescription = product.title,
                modifier = Modifier
                    .fillMaxWidth()
                    .height(180.dp)
            )
            Spacer(modifier = Modifier.height(8.dp))
            Text(
                text = product.title,
                style =
MaterialTheme.typography.titleMedium,
                maxLines = 2,
                overflow = TextOverflow.Ellipsis,
                color = Color.Black
            )
            Spacer(modifier = Modifier.height(8.dp))

```

```
            Text(text = "$${product.price}", style =
MaterialTheme.typography.bodyMedium, color =
Color.Black)
            Spacer(modifier = Modifier.height(8.dp))
            Button(onClick = onDetailClick) {
                Text("Detail")
            }
            Button(onClick = {
                val searchQuery =
Uri.encode(product.title)
                val url =
"https://shopee.co.id/search?keyword=$searchQuery"
                val intent = Intent(Intent.ACTION_VIEW,
Uri.parse(url))
                context.startActivity(intent)
            }) {
                Text("Buy Now on Website")
            }
        }
    }
}

@Composable
fun DetailScreen(productId: Int, viewModel:
HomeViewModel) {
    val product =
viewModel.products.collectAsState().value.find { it.id
== productId }

    val isLoading by
viewModel.isLoading.collectAsState()
    val error by viewModel.error.collectAsState()

    Scaffold(
        topBar = {
            TopAppBar(
                title = { Text("Product Detail") }
            )
        }
    ) { padding ->
        Box(
            modifier = Modifier
                .padding(padding)
                .fillMaxSize()
                .background(Color.White),
            contentAlignment = Alignment.Center
        ) {
            when {
                isLoading -> {

```

```
                        CircularProgressIndicator()
                    }
                    error != null -> {
                        Text(
                            text = error ?: "Unknown
error",
                            color =
MaterialTheme.colorScheme.error
                        )
                    }
                    product != null -> {
                        Column(
                            modifier =
Modifier.padding(16.dp)
                                .fillMaxSize()
                                .background(Color.White),
                            verticalArrangement =
Arrangement.spacedBy(8.dp)
                        ) {
                            AsyncImage(
                                model = product.image,
                                contentDescription =
product.title,
                                modifier = Modifier
                                    .fillMaxWidth()
                                    .height(180.dp)
                            )
                            Text("Name: ${product.title}",
style = MaterialTheme.typography.titleMedium, color =
Color.Black)
                            Text("Price:
$${product.price}", color = Color.Black)
                            Text("Category:
${product.category}", color = Color.Black)
                            Text("Description:
${product.description}", color = Color.Black)
                            Text("Rating:
${product.rating}", color = Color.Black   )
                        }
                    }
                    else -> {
                        Text("No product data", color =
Color.Black)
                    }
                }
            }
        }
    }
}
```

7. ui/home/HomeViewModel.kt

Tabel 19 Source Code ui/home/HomeViewModel.kt Modul 5

```
11  package com.example.modul_5
12
13  import android.os.Bundle
14  import androidx.activity.ComponentActivity
15  import androidx.activity.compose.setContent
16  import androidx.activity.enableEdgeToEdge
17  import androidx.compose.foundation.layout.fillMaxSize
18  import androidx.compose.foundation.layout.padding
19  import androidx.compose.material3.Scaffold
20  import androidx.compose.ui.Modifier
21  import androidx.lifecycle.viewmodel.compose.viewModel
22  import com.example.modul_5.ui.home.HomeNavHost
23  import com.example.modul_5.ui.theme.Modul_5Theme
24  import com.example.modul_5.ui.home.HomeViewModel
25
26  class MainActivity : ComponentActivity() {
27      override fun onCreate(savedInstanceState: Bundle?) {
28          super.onCreate(savedInstanceState)
29          enableEdgeToEdge()
30          setContent {
31              Modul_5Theme {
32                  Scaffold(modifier =
33  Modifier.fillMaxSize()) { innerPadding ->
34
35                      val homeViewModel: HomeViewModel =
36  viewModel()
37
38                      HomeNavHost(
39                          viewModel = homeViewModel,
40                          modifier =
41  Modifier.padding(innerPadding)
42                      )
43                  }
44              }
45          }
46  }
```

8. ui/theme/Theme.kt

Tabel 20 Source Code ui/theme/Theme.kt Modul 5

```
11  package com.example.modul_5.ui.theme
12
13  import android.app.Activity
```

```
import android.os.Build
import androidx.compose.foundation.isSystemInDarkTheme
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.darkColorScheme
import androidx.compose.material3.dynamicDarkColorScheme
import androidx.compose.material3.dynamicLightColorScheme
import androidx.compose.material3.lightColorScheme
import androidx.compose.runtime.Composable
import androidx.compose.ui.platform.LocalContext

private val DarkColorScheme = darkColorScheme(
    primary = Purple80,
    secondary = PurpleGrey80,
    tertiary = Pink80
)

private val LightColorScheme = lightColorScheme(
    primary = Purple40,
    secondary = PurpleGrey40,
    tertiary = Pink40

    /* Other default colors to override
    background = Color(0xFFFFFBFE),
    surface = Color(0xFFFFFBFE),
    onPrimary = Color.White,
    onSecondary = Color.White,
    onTertiary = Color.White,
    onBackground = Color(0xFF1C1B1F),
    onSurface = Color(0xFF1C1B1F),
    */
)

@Composable
fun Modul_5Theme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    // Dynamic color is available on Android 12+
    dynamicColor: Boolean = true,
    content: @Composable () -> Unit
) {
    val colorScheme = when {
        dynamicColor && Build.VERSION.SDK_INT >=
Build.VERSION_CODES.S -> {
            val context = LocalContext.current
            if (darkTheme)
dynamicDarkColorScheme(context) else
dynamicLightColorScheme(context)
        }

        darkTheme -> DarkColorScheme

```

```
63          else -> LightColorScheme
64      }
65
66      MaterialTheme(
67          colorScheme = colorScheme,
68          typography = Typography,
69          content = content
70      )
}
```

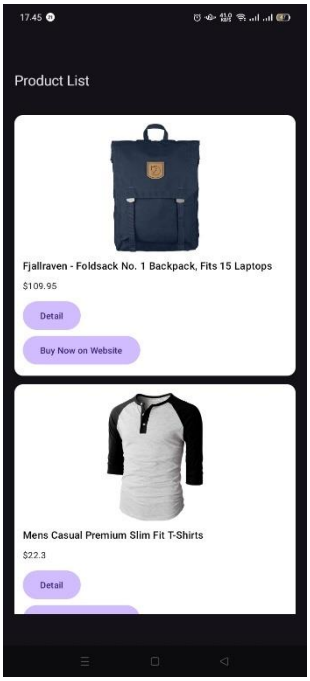
9. MainActivity.kt

Tabel 21 Source Code MainActivity.kt Modul 5

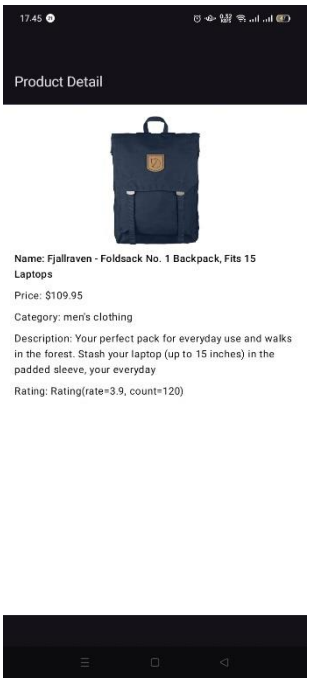
```
1   package com.example.modul_5
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.enableEdgeToEdge
7   import androidx.compose.foundation.layout.fillMaxSize
8   import androidx.compose.foundation.layout.padding
9   import androidx.compose.material3.Scaffold
10  import androidx.compose.ui.Modifier
11  import androidx.lifecycle.viewmodel.compose.viewModel
12  import com.example.modul_5.ui.home.HomeNavHost
13  import com.example.modul_5.ui.theme.Modul_5Theme
14  import com.example.modul_5.ui.home.HomeViewModel
15
16  class MainActivity : ComponentActivity() {
17      override fun onCreate(savedInstanceState: Bundle?) {
18          super.onCreate(savedInstanceState)
19          enableEdgeToEdge()
20          setContent {
21              Modul_5Theme {
22                  Scaffold(modifier =
23  Modifier.fillMaxSize()) { innerPadding ->
24
25                      val homeViewModel: HomeViewModel =
26  viewModel()
27
28                      HomeNavHost(
29                          viewModel = homeViewModel,
30                          modifier =
31  Modifier.padding(innerPadding)
32                      )
33                  }
34              }
          }
```

```
35          }
36  }
```

B. Output Program



Gambar 22 Screenshot Tampilan Halaman Awal Aplikasi Modul 5



Gambar 23 Screenshot Tampilan Halaman Detail Product Aplikasi Modul 5

Gambar 24 Screenshot Tombol "Buy On Website" di Aplikasi Ditekan Modul 5

C. Pembahasan

1. model/Produt.kt

Pada file ini kita membuat sebuah data class dengan nama Product yang ditandai dengan anotasi @Serializable, yang artinya class ini bisa digunakan dalam proses serialisasi atau konversi data (misalnya JSON). Class Product ini berisi beberapa properti penting seperti id, title, price, description, category, image, dan rating. Semua properti ini mewakili atribut dari sebuah produk secara lengkap.

Selain itu, di file ini juga terdapat class Rating yang juga diberi anotasi @Serializable. Class ini merupakan bagian dari properti Product, dan berfungsi untuk menyimpan nilai rating dan jumlah rating dari produk tersebut. File ini nantinya akan digunakan saat kita mengambil atau menampilkan data produk, baik dari API ataupun dari repository lokal.

2. remote/ProductApiService.kt

Pada file ini kita membuat sebuah interface dengan nama ProductApiService yang berfungsi sebagai jembatan untuk berkomunikasi dengan API eksternal menggunakan Retrofit. Di dalamnya terdapat satu fungsi bernama getProducts() yang diberi anotasi @GET("products"). Anotasi ini menandakan bahwa fungsi ini akan melakukan request HTTP GET ke endpoint products dan akan mengembalikan daftar data bertipe List<Product>.

Fungsi getProducts() bersifat suspend, artinya hanya bisa dipanggil dari coroutine, karena proses pengambilan data dari internet bersifat asynchronous. File ini akan sangat penting saat menghubungkan aplikasi dengan data dari server atau API eksternal.

3. remote/RetrofitInstance.kt

Pada file ini kita membuat sebuah object dengan nama RetrofitInstance yang berfungsi untuk mengatur dan menyediakan instance Retrofit yang akan digunakan untuk mengakses API. Di dalamnya, kita menggunakan konfigurasi Json dari Kotlin Serialization yang disesuaikan dengan properti ignoreUnknownKeys = true, yang berarti jika ada data dari API yang tidak sesuai dengan struktur data di aplikasi, maka akan diabaikan agar tidak menimbulkan error.

Selanjutnya, kita membuat properti api bertipe ProductApiService yang diinisialisasi secara lazy. Di dalamnya, kita menggunakan Retrofit.Builder() dengan baseUrl yang mengarah ke https://fakestoreapi.com/ dan menambahkan ConverterFactory dari Kotlinx Serialization agar data JSON bisa dikonversi otomatis ke objek Kotlin. File ini menjadi kunci utama untuk menghubungkan aplikasi dengan API, dan akan digunakan saat kita ingin mengambil data produk secara online melalui Retrofit.

4. repository/ProductRepository.kt

Pada file ini kita membuat sebuah class dengan nama ProductRepository yang berfungsi sebagai pengelola data dari sumber eksternal, dalam hal ini adalah API. Di dalam class ini terdapat satu fungsi utama getProducts() yang akan mengembalikan data berupa Flow<Result<List<Product>>>. Ini berarti data produk akan dikirimkan secara asynchronous dan bisa ditangani dengan cara yang reaktif menggunakan Flow.

Fungsi ini mengambil data dari RetrofitInstance.api.getProducts() yang telah didefinisikan sebelumnya, lalu menggunakan emit() untuk mengirimkan hasilnya. Jika data berhasil diambil, maka akan dikirim dalam bentuk Result.success(products). Namun, jika terjadi kesalahan seperti error jaringan atau parsing, maka akan dikirim dalam bentuk Result.failure(e). Logcat juga digunakan untuk mencatat keberhasilan atau kegagalan proses pengambilan data, yang membantu dalam proses debugging.

File ini menjadi penghubung antara lapisan data (API) dan ViewModel, sehingga pemisahan tanggung jawab antar bagian dalam arsitektur aplikasi tetap terjaga.

5. ui/detail/DetailScreen.kt

Pada file ini kita membuat tampilan detail produk yang diberi nama DetailScreen. Fungsi ini merupakan komponen Composable yang digunakan untuk menampilkan informasi lengkap dari salah satu produk berdasarkan ID-nya. Fungsi ini menerima dua parameter, yaitu productId yang merepresentasikan ID produk yang ingin ditampilkan, dan viewModel dari HomeViewModel yang akan digunakan untuk mengambil state produk dari ViewModel. Data yang diamati mencakup tiga hal utama: products yang merupakan daftar seluruh produk, isLoading untuk mengetahui apakah data sedang dimuat, serta error yang akan menampilkan pesan jika terjadi kesalahan saat pengambilan data.

Struktur UI-nya dibangun menggunakan Scaffold yang dilengkapi dengan TopAppBar sebagai header dengan judul "Detail Produk". Kemudian, isi dari tampilan akan menyesuaikan kondisi yang sedang terjadi. Jika data masih dimuat, maka akan ditampilkan CircularProgressIndicator. Jika terjadi kesalahan, maka akan muncul pesan kesalahan yang ditandai dengan warna error. Jika produk berhasil ditemukan berdasarkan productId, maka fungsi ProductDetailContent akan dipanggil untuk menampilkan informasi produk seperti nama, harga, kategori, dan deskripsi secara terstruktur dalam bentuk kolom. Namun jika produk tidak ditemukan, akan muncul teks informasi bahwa produk tidak tersedia.

6. ui/home/HomeScreen.kt

Pada file ini kita membuat beberapa fungsi composable utama untuk menampilkan tampilan utama aplikasi. Fungsi HomeNavHost berfungsi sebagai pengatur navigasi antar layar dengan menggunakan NavHost dan NavController. Di dalamnya terdapat dua route utama yaitu "home" yang menampilkan daftar produk melalui HomeScreen, serta "detail/{productId}" yang menampilkan detail produk tertentu berdasarkan productId melalui DetailScreen.

Fungsi HomeScreen menampilkan daftar produk dalam bentuk list menggunakan LazyColumn. Data produk diambil dari HomeViewModel yang dipantau melalui collectAsState(). Pada tampilan ini terdapat kondisi loading yang ditandai dengan CircularProgressIndicator, kondisi error yang menampilkan pesan error, dan kondisi

sukses yang menampilkan daftar produk menggunakan ProductCard. Setiap item produk dapat diklik untuk menuju detail produk atau membuka halaman website Shopee untuk membeli produk tersebut melalui browser.

ProductCard adalah komponen UI yang merepresentasikan satu produk dalam bentuk kartu yang berisi gambar produk, judul, harga, dan dua tombol aksi yaitu tombol untuk melihat detail produk dan tombol untuk membuka website pembelian. Tombol pembelian mengarahkan pengguna ke browser dengan pencarian produk di Shopee menggunakan Intent.ACTION_VIEW.

Fungsi DetailScreen menampilkan rincian lengkap dari produk berdasarkan productId yang diterima. Data produk juga diambil dari HomeViewModel dan dipantau status loading serta error-nya. Jika data berhasil didapat, detail produk seperti gambar, nama, harga, kategori, deskripsi, dan rating akan ditampilkan secara rapi. Jika produk tidak ditemukan atau terjadi error, pesan yang sesuai akan ditampilkan. File ini sangat penting sebagai penghubung antara data produk dan tampilan UI di aplikasi dengan memanfaatkan Jetpack Compose serta Navigation Compose untuk pengalaman navigasi yang lancar dan interaktif.

7. ui/home/HomeViewModel.kt

Pada file ini dibuat sebuah kelas HomeViewModel yang berperan sebagai jembatan antara data repository dan UI. Kelas ini menggunakan ViewModel dari Android Architecture Components dan mengelola state aplikasi dengan memanfaatkan StateFlow dan MutableStateFlow untuk mengawasi perubahan data produk, status loading, serta error. Di dalam HomeViewModel terdapat tiga properti utama yang dibungkus StateFlow: _products yang menyimpan daftar produk dalam bentuk list, _isLoading yang menandakan status pemuatan data sedang berlangsung, dan _error yang menyimpan pesan error jika terjadi kegagalan dalam pengambilan data.

Pada inisialisasi kelas, fungsi fetchProducts() langsung dipanggil untuk memulai pengambilan data produk dari ProductRepository. Fungsi ini menggunakan viewModelScope.launch agar proses fetching berjalan dalam coroutine tanpa

mengganggu thread utama. Ketika data berhasil diambil, products diperbarui dengan data terbaru dan error diset null. Jika terjadi kegagalan, daftar produk dikosongkan dan pesan error ditampilkan. File ini sangat penting untuk mengatur alur data dan status aplikasi sehingga UI dapat merespons perubahan data secara reaktif dan efisien dengan menggunakan Jetpack Compose.

8. ui/theme/Theme.kt

File ini bertugas mengatur tema visual aplikasi menggunakan Jetpack Compose Material3, termasuk dukungan untuk mode gelap (dark mode) dan mode terang (light mode). Di dalamnya didefinisikan dua skema warna utama, yaitu DarkColorScheme untuk tema gelap dan LightColorScheme untuk tema terang, dengan warna-warna primer, sekunder, dan tersier yang sudah ditentukan.

Fungsi composable Modul_5Theme adalah pusat pengaturan tema. Fungsi ini secara otomatis mendeteksi apakah perangkat sedang menggunakan mode gelap melalui isSystemInDarkTheme(), sehingga aplikasi bisa menyesuaikan tampilannya mengikuti preferensi pengguna tanpa pengaturan manual. Selain itu, untuk perangkat Android versi 12 ke atas, aplikasi memanfaatkan fitur dynamic color scheme yang mengambil warna tema dari wallpaper perangkat, membuat tampilan aplikasi lebih menyatu dan personal.

9. MainActivity.kt

File ini merupakan entry point utama aplikasi yang mengatur tampilan dan alur navigasi menggunakan Jetpack Compose. Di dalamnya terdapat class MainActivity yang mewarisi ComponentActivity. Pada fungsi onCreate, pertama-tama dipanggil enableEdgeToEdge() agar aplikasi bisa menampilkan konten hingga ke tepi layar, memberikan tampilan yang lebih modern dan immersive.

Selanjutnya, dengan setContent aplikasi mulai menggunakan tema yang sudah dibuat di Modul_5Theme sehingga tampilan mengikuti pengaturan warna dan tipografi yang konsisten, termasuk dukungan dark mode.

Di dalam tema tersebut, digunakan Scaffold sebagai struktur dasar UI yang menyediakan area standar seperti top bar, bottom bar, dan konten utama. Padding dari scaffold ini dioper ke konten agar tata letak responsif terhadap elemen UI lainnya.

ViewModel HomeViewModel diinisialisasi menggunakan Compose viewModel() yang memastikan data produk dikelola secara lifecycle-aware dan bisa digunakan di seluruh composable.

Terakhir, fungsi HomeNavHost dipanggil untuk mengatur navigasi antar layar (home dan detail produk) dengan ViewModel yang sudah tersedia, sehingga aplikasi dapat menampilkan daftar produk dan detailnya secara dinamis.

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/KunyitAlami/Praktikum-Pemrograman-Mobile-I-Ghani-Mudzakir.git