LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Muhammad Rizki Saputra **NIM. 2310817310014**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

MEI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Muhammad Rizki Saputra
NIM : 2310817310014

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

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

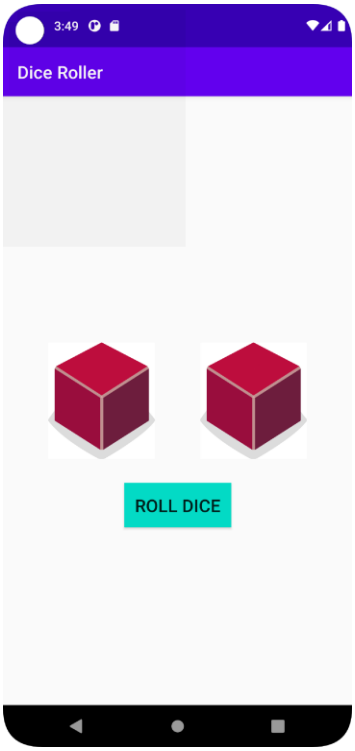
MODUL 1 : Android Basic with Kotlin

SOAL 1

Soal Praktikum:

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

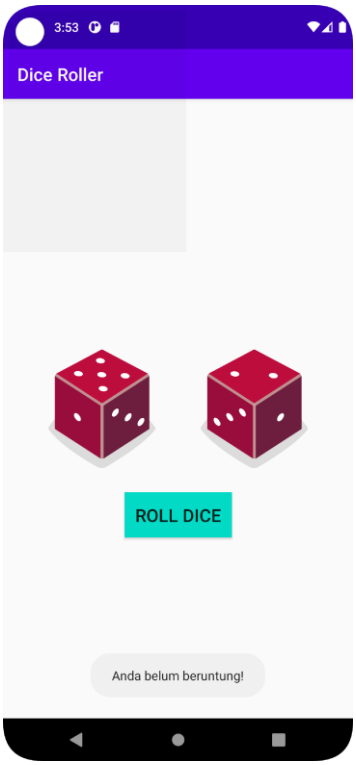
1. Tampilan awal aplikasi setelah dijalankan akan menampilkan 2 buah dadu kosong seperti dapat dilihat pada Gambar 1.



Gambar 1. Soal 1 Modul 1

2. Setelah user menekan tombol “Roll Dice” maka masing-masing dadu akan memunculkan sisi dadu masing-masing dengan angka antara 1 s/d 6. Apabila user mendapatkan nilai dadu yang berbeda antara Dadu 1 dengan Dadu 2 maka akan

menampilkan pesan “Anda belum beruntung!” seperti dapat dilihat pada Gambar 2.

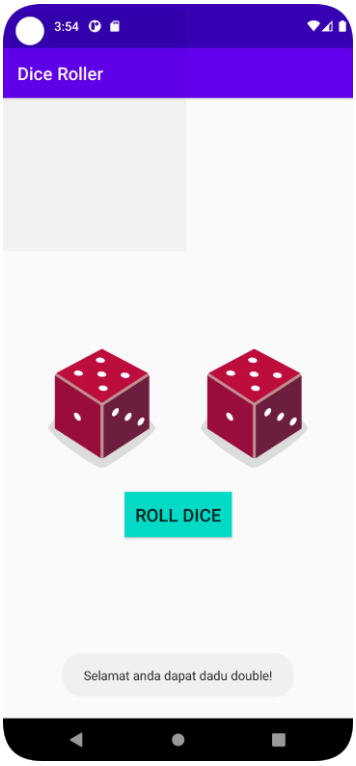


Gambar 2. Soal 1 Modul 1

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

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. Soal 1 Modul 1

A. Source Code

- MainActivity.kt

Table 1. SourceCode MainActivity Soal 1 Modul 1

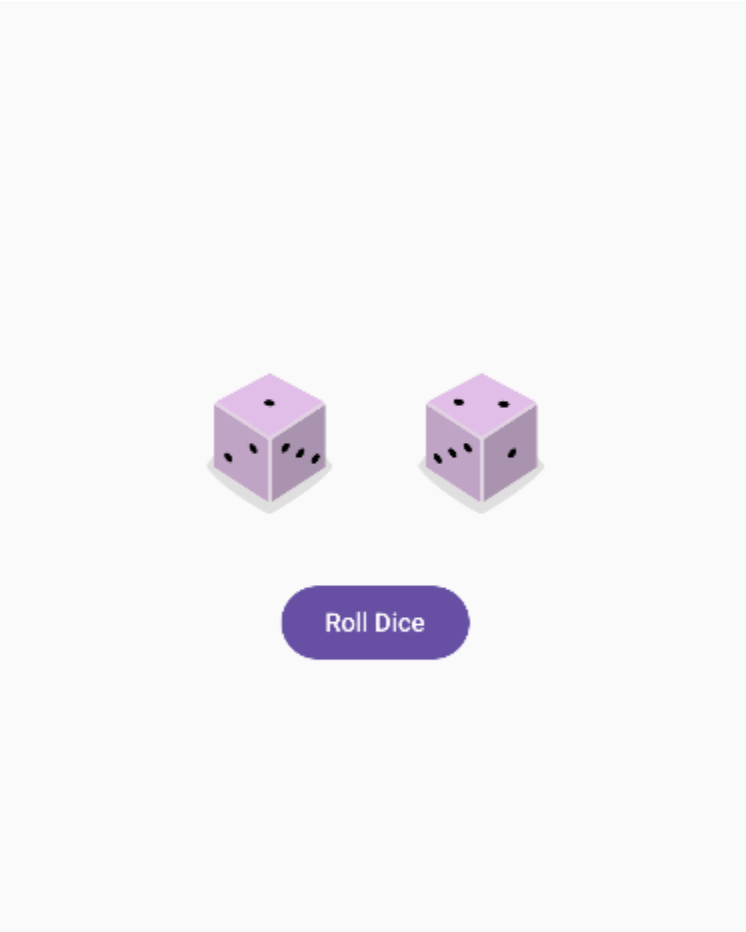
```
1  package com.example.modul1
2
3  import android.os.Bundle
4  import android.widget.Toast
5  import androidx.activity.ComponentActivity
6  import androidx.activity.compose.setContent
7  import androidx.compose.foundation.Image
8  import androidx.compose.foundation.layout.*
9  import androidx.compose.material3.*
10
```

```
11 import androidx.compose.runtime.*
12 import androidx.compose.ui.Alignment
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.platform.LocalContext
15 import androidx.compose.ui.res.painterResource
16 import androidx.compose.ui.unit.dp
17 import com.example.modul1.R
18 import kotlin.random.Random
19
20 class MainActivity : ComponentActivity() {
21     override fun onCreate(savedInstanceState: Bundle?) {
22         super.onCreate(savedInstanceState)
23         setContent {
24             DiceRollApp()
25         }
26     }
27 }
28
29 @Composable
30 fun DiceRollApp() {
31     var dice1 by remember { mutableStateOf(0) }
32     var dice2 by remember { mutableStateOf(0) }
33     val context = LocalContext.current
34
35     LaunchedEffect(dice1, dice2) {
36         if (dice1 != 0 && dice2 != 0) {
37             if (dice1 == dice2) {
38                 Toast.makeText(context, "Selamat anda dapat dadu double!",
39 Toast.LENGTH_SHORT).show()
40             } else {
41                 Toast.makeText(context, "Anda belum beruntung!",
```

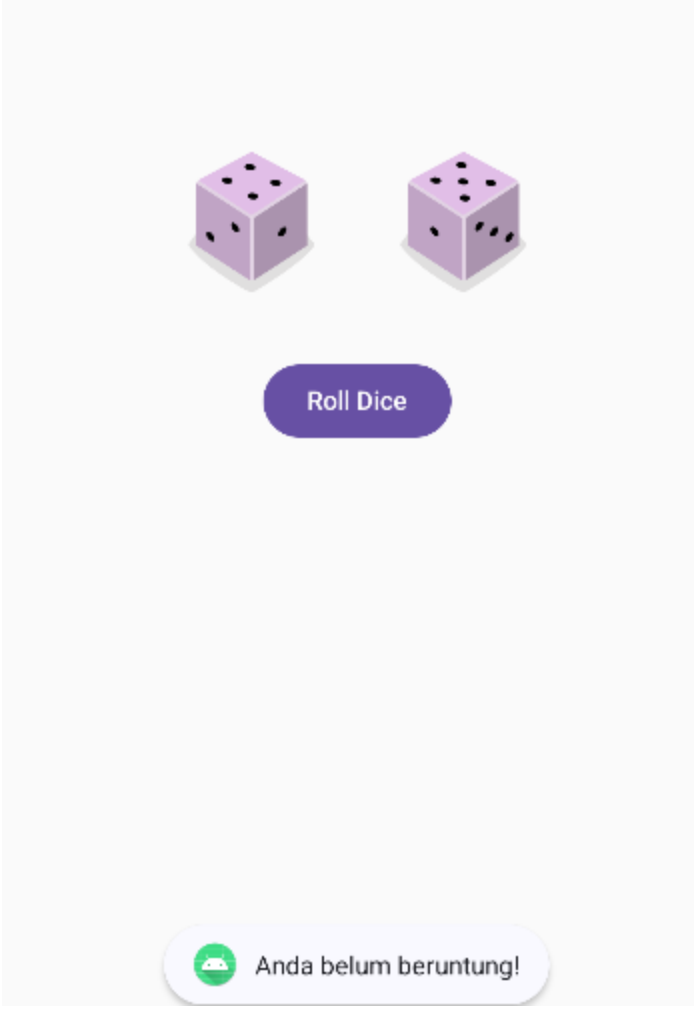
```
42  Toast.LENGTH_SHORT).show()
43              }
44          }
45      }
46
47
48      Column(
49          modifier = Modifier
50              .fillMaxSize()
51              .padding(16.dp),
52          verticalArrangement = Arrangement.Center,
53          horizontalAlignment = Alignment.CenterHorizontally
54      ) {
55          Row(
56              horizontalArrangement = Arrangement.spacedBy(16.dp)
57          ) {
58              Image(
59                  painter = painterResource(id = getDiceImage(dice1)),
60                  contentDescription = "Dadu 1",
61                  modifier = Modifier.size(100.dp)
62              )
63              Image(
64                  painter = painterResource(id = getDiceImage(dice2)),
65                  contentDescription = "Dadu 2",
66                  modifier = Modifier.size(100.dp)
67              )
68          }
69
70          Spacer(modifier = Modifier.height(24.dp))
71
72          Button(onClick = {
```

```
73              dice1 = Random.nextInt(1, 7)
74              dice2 = Random.nextInt(1, 7)
75          }) {
76              Text(text = "Roll Dice")
77          }
78
79          Spacer(modifier = Modifier.height(16.dp))
80
81      }
82  }
83
84  fun getDiceImage(value: Int): Int {
85      return when (value) {
86          1 -> R.drawable.dice_1
87          2 -> R.drawable.dice_2
88          3 -> R.drawable.dice_3
89          4 -> R.drawable.dice_4
90          5 -> R.drawable.dice_5
92          6 -> R.drawable.dice_6
93          else -> R.drawable.dice_0
93      }
94  }
```

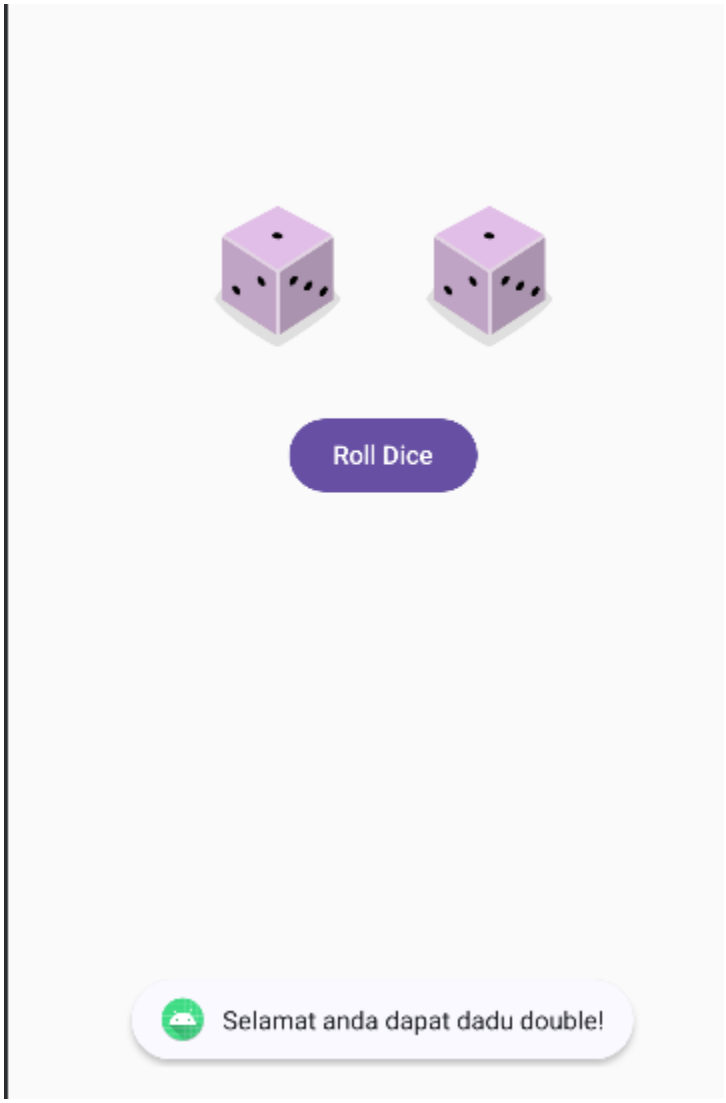
B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 Modul 1



Gambar 5. Screenshot Hasil Jawaban Soal 1 Modul 1



Gambar 6. Screenshot Hasil Jawaban Soal 1 Modul 1

C. Pembahasan

MainActivity.kt:

- Baris 1: package com.example.modul1 Menentukan paket tempat file Kotlin ini berada. Ini seperti folder khusus dalam proyek Android.
- Baris 3–17 Mengimpor class dan fungsi dari Android dan Jetpack Compose yang diperlukan, seperti Bundle, Toast, layouting (Column, Row, dll), dan utilitas Compose lainnya.
- Baris 19 class MainActivity : ComponentActivity() Mendeklarasikan MainActivity sebagai class utama dan mewarisi ComponentActivity, yang menjadi titik awal UI Compose dijalankan.
- Baris 7–11 Fungsi onCreate() adalah tempat inisialisasi saat activity dibuat. Di dalamnya, fungsi setContent { DiceRollApp() } dipanggil untuk menampilkan UI yang dibuat dengan Compose.
- Baris 20: @Composable Menandakan bahwa fungsi DiceRollApp() adalah fungsi composable, yaitu bagian dari UI yang dapat dirender dan direkomposisi secara otomatis.
- Baris 31–32 Variabel dice1 dan dice2 disimpan dengan remember dan mutableStateOf(0). Ini memungkinkan Compose melacak dan merespons perubahan nilainya di UI.
- Baris 33: val context = LocalContext.current Mendapatkan konteks aplikasi untuk keperluan menampilkan Toast.
- Baris 36–47 Blok LaunchedEffect(dice1, dice2) berjalan setiap kali dice1 atau dice2 berubah. Jika keduanya bernilai sama (double), Toast akan menampilkan pesan "Selamat anda dapat dadu double!", jika tidak "Anda belum beruntung!".
- Baris 50–59 Column(...) adalah layout vertikal untuk menyusun elemen UI dari atas ke bawah. Modifier fillMaxSize() memenuhi layar, dan padding(16.dp) memberi jarak di semua sisi.
- Baris 57–59 Row(...) menampilkan dua elemen berdampingan (gambar dadu) dengan jarak horizontal 16dp. Sangat cocok untuk menampilkan dua dadu sejajar.

- Baris 60–69 Dua Image(...) digunakan untuk menampilkan gambar dadu berdasarkan hasil angka. Gambar diambil dari resource dengan fungsi getDiceImage().
- Baris 75 Spacer(modifier = Modifier.height(24.dp))Memberi jarak vertikal antara gambar dan tombol.
- Baris 77–79 Tombol Button(...) digunakan untuk melempar dadu. Saat diklik, dice1 dan dice2 diisi dengan angka acak dari 1–6 menggunakan Random.nextInt(1, 7).
- Baris 82: Text(text = "Roll Dice")Label pada tombol yang ditampilkan kepada pengguna.
- Baris 85: Spacer(modifier = Modifier.height(16.dp))Menambah jarak di bagian bawah tombol agar UI terlihat lebih rapi.
- Baris 90–99 Fungsi getDiceImage(value: Int): Int berfungsi mengembalikan resource ID dari gambar dadu berdasarkan nilai yang diberikan. Jika nilainya tidak valid (di luar 1–6), maka akan mengembalikan dice_0.

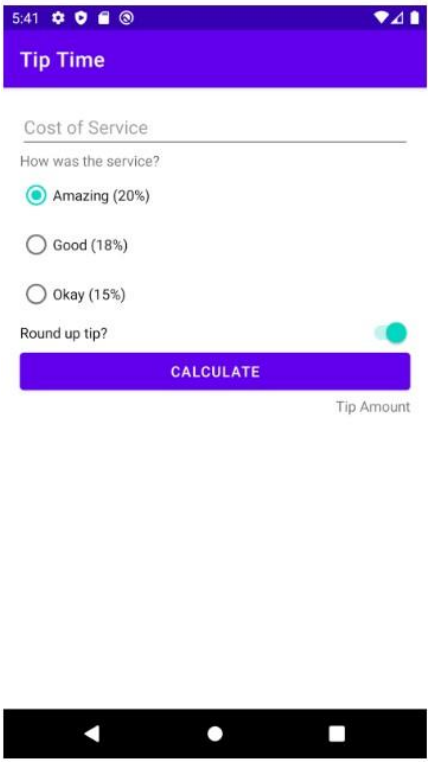
MODUL 2 : ANDROID LAYOUT

SOAL 1

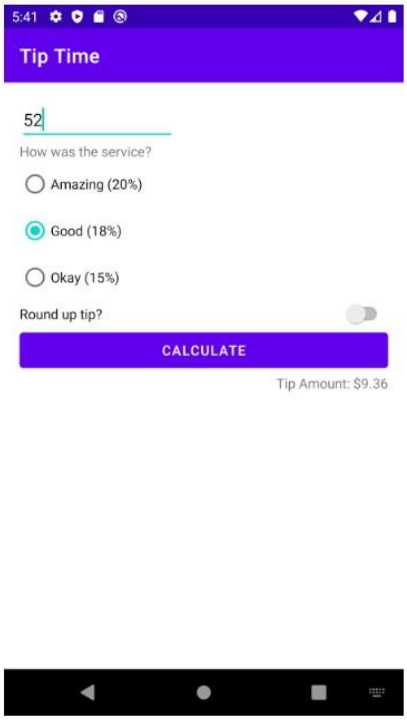
Soal Praktikum:

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

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



Gambar 7. Soal 1 Modul 2



Gambar 8. Soal 1 Modul 2

A. Source Code

- MainActivity.kt

Table 2. Sourcecode MainActivity Soal 1 Modul 2

```
1  package com.example.modul2
2
3  import android.os.Bundle
4  import androidx.activity.ComponentActivity
5  import androidx.activity.compose.setContent
6  import androidx.compose.foundation.background
7  import androidx.compose.foundation.layout.*
8  import androidx.compose.foundation.text.KeyboardOptions
9  import androidx.compose.material3.*
10 import androidx.compose.material3.RadioButton
11 import androidx.compose.runtime.*
12 import androidx.compose.ui.Alignment
13 import androidx.compose.ui.Modifier
14 import androidx.compose.ui.graphics.Color
15 import androidx.compose.ui.text.input.KeyboardType
16 import androidx.compose.ui.text.style.TextAlign
17 import androidx.compose.ui.unit.dp
18 import androidx.compose.ui.unit.sp
19 import
20 androidx.compose.foundation.shape.RoundedCornerShape
21 import kotlin.math.ceil
22
23 class MainActivity : ComponentActivity() {
24     override fun onCreate(savedInstanceState: Bundle?) {
25         super.onCreate(savedInstanceState)
26         setContent {
```

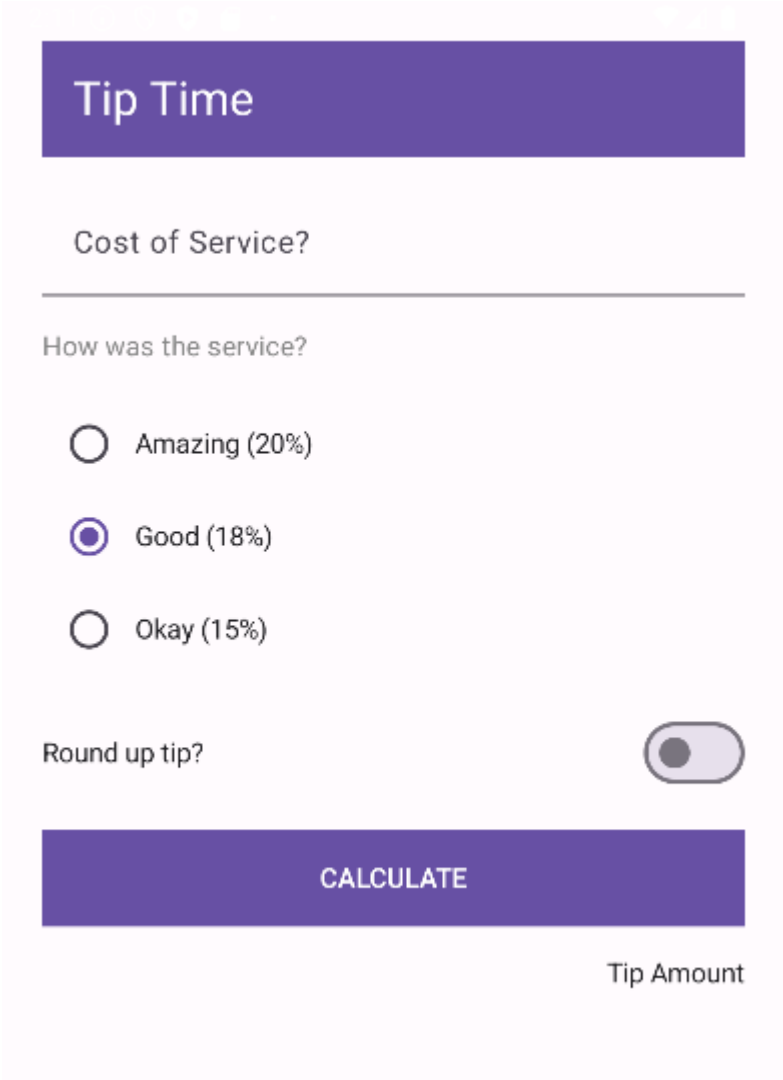
```
27              Surface(modifier = Modifier.fillMaxSize()) {
28                  TipCalculatorScreen()
29              }
30          }
31      }
32  }
33
34  @OptIn(ExperimentalMaterial3Api::class)
35  @Composable
36  fun TipCalculatorScreen() {
37      var costInput by remember { mutableStateOf("") }
38      var tipPercent by remember { mutableStateOf(0.18) }
39      var roundUp by remember { mutableStateOf(false) }
40      var tipResult by remember { mutableStateOf("") }
41
42      val cost = costInput.toDoubleOrNull() ?: 0.0
43      var tip = cost * tipPercent
44      if (roundUp) tip = ceil(tip)
45
46      Column(
47          modifier = Modifier
48              .fillMaxSize()
49              .padding(24.dp),
50          verticalArrangement = Arrangement.spacedBy(16.dp)
51      ) {
51          Text(
52              "Tip Time",
53              fontSize = 24.sp,
54              color = Color.White,
55              modifier = Modifier
56                  .fillMaxWidth()
```

```
57
58      .background(MaterialTheme.colorScheme.primary)
59                      .padding(16.dp)
60              )
61
62              TextField(
63                  value = costInput,
64                  onValueChange = { costInput = it },
65                  placeholder = { Text("Cost of Service?") },
66      keyboardOptions = KeyboardOptions(keyboardType =
67      KeyboardType.Number),
68                  modifier = Modifier.fillMaxWidth(),
69                  colors =
70      TextFieldDefaults.textFieldColors(containerColor =
71      Color.Transparent)
72              )
73
74              Text(
75                  text = "How was the service?",
76                  style = MaterialTheme.typography.bodyMedium,
77                  color = Color.Gray
78              )
79
80              Column {
81                  TipOption("Amazing (20%)", 0.20, tipPercent) {
82      tipPercent = it }
83                  TipOption("Good (18%)", 0.18, tipPercent) {
84      tipPercent = it }
85                  TipOption("Okay (15%)", 0.15, tipPercent) {
86      tipPercent = it }
87              }
```

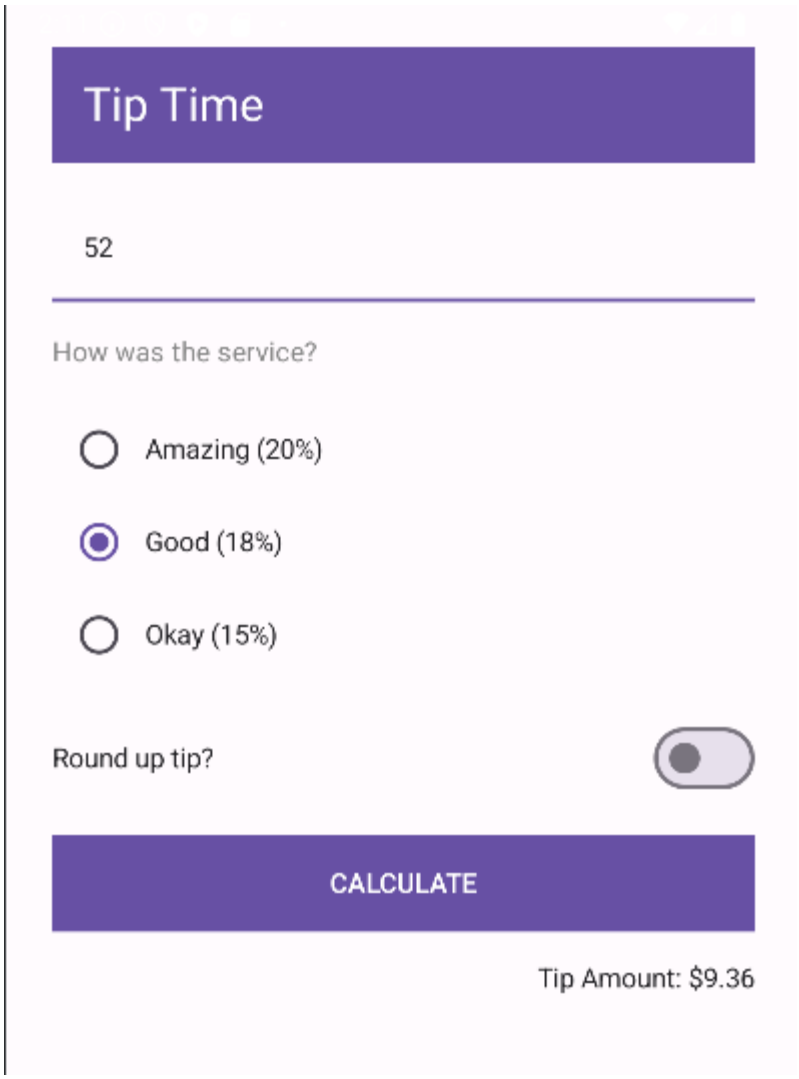
```
88
89          Row(
90              modifier = Modifier.fillMaxWidth(),
91              verticalAlignment =
92  Alignment.CenterVertically,
93              horizontalArrangement =
94  Arrangement.SpaceBetween
95          ) {
96              Text("Round up tip?")
97              Switch(
98                  checked = roundUp,
99                  onCheckedChange = { roundUp = it }
100             )
101         }
102
103         Button(
104             onClick = {
105                 tipResult = "Tip Amount:
106 $${"%.2f".format(tip)}"
107             },
108             modifier = Modifier
109                 .fillMaxWidth()
110                 .height(50.dp),
111             colors =
112 ButtonDefaults.buttonColors(containerColor =
113 MaterialTheme.colorScheme.primary),
114             shape = RoundedCornerShape(0.dp)
115         ) {
116             Text("CALCULATE", color = Color.White)
117         }
118
```

```
        Text(
    text = if (tipResult.isEmpty()) "Tip Amount" else
tipResult,
    modifier = Modifier.fillMaxWidth(),
    textAlign = TextAlign.End
)
    }
}

@Composable
fun TipOption(text: String, value: Double, selected:
Double, onSelect: (Double) -> Unit) {
    Row(verticalAlignment = Alignment.CenterVertically) {
        RadioButton(
            selected = selected == value,
            onClick = { onSelect(value) }
        )
        Text(text)
    }
}
```

B. Output Program



Gambar 9. Screenshot Hasil Jawaban Soal 1 Modul 2



Gambar 10. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

MainActivity.kt:

- Baris 1: package com.example.modul1 Menentukan paket tempat file Kotlin ini berada. Ini seperti folder khusus dalam proyek Android.
- Baris 3–20 Mengimpor class dan fungsi dari Android dan Jetpack Compose yang diperlukan, seperti Bundle, Toast, layouting (Column, Row, dll), dan utilitas Compose lainnya
- Baris 22 sampai 27, bertujuan untuk mendeklarasikan class MainActivity yang merupakan turunan dari ComponentActivity, disana terdapat onCrate sebuah method yang berguna untuk jika dipanggil saat Activity dibuat, lalu setContent, digunakan untuk mengisi tampilan menggunakan Jetpack Compose, dan Surface sebagai container utama.
- Baris 34 @OptIn(ExperimentalMaterial3Api::class) Mengizinkan penggunaan fitur Material 3 yang masih dalam tahap eksperimen (belum stabil sepenuhnya).
- Baris 37–40 terdapat deklarasi variable yang Dimana, costInput: untuk menyimpan input biaya layanan, tipPercent: untuk menyimpan persentase tip yang dipilih, roundUp: boolean untuk menentukan apakah hasil tip dibulatkan ke atas. Dan tipResult: untuk menyimpan hasil teks yang akan ditampilkan ke pengguna.
- Baris 42 sampai 45 terdapat proses penghitungan nilai tip yaitu, cost diisi dengan konversi costInput menjadi angka, jika gagal maka diisi 0.0 kemudian tip dihitung sebagai cost * tipPercent Dan jika roundUp bernilai true, maka nilai tip dibulatkan ke atas menggunakan ceil().
- Baris 47 sampai 51, membuat susuan layout menggunakan column dengan cara, fillMaxSize() membuat layout memenuhi seluruh layar. Kemudian padding(24.dp) memberi jarak dari pinggir layar sebesar 24dp. Dan verticalArrangement = Arrangement.spacedBy(16.dp) mengatur jarak antar elemen sebesar 16dp.
- Baris 53 sampai 61 menampilkan teks judul “Tip Time” berukuran font 24sp, lalu menggunakan background warna utama dengan cara

(MaterialTheme.colorScheme.primary), dan teks dipusatkan dan diberi padding 16dp di sekelilingnya

- Baris 64 sampai 75, akan menampilkan TextFIeld untuk memasukkan biaya, keyboardOptions diatur oleh KeyboardType.Number agar hanya angka yang bisa diinput. Dan untuk warna latar belakang TextField dibuat transparan. Lalu placeholder bertuliskan "Cost of Service?".
- Baris 78 sampai 81, Menampilkan teks kecil "How was the service?" di bawah input biaya. Dan menggunakan style bodyMedium. Untuk warnanya teks dibuat abu-abu (Color.Gray).
- Baris 84 sampai 91, Menampilkan tiga opsi RadioButton untuk memilih persentase tip:

  "Amazing (20%)"
  "Good (18%)"
  "Okay (15%)"
  Setiap opsi memanggil fungsi TipOption().
- Baris 99 sampai 103, berguna untuk menampilakn row yang berisi teks "Round up tip?" di sisi kiri. Switch di sisi kanan untuk mengaktifkan atau menonaktifkan pembulatan ke atas. Dan Switch akan mengubah nilai roundUp saat diklik.
- Baris 106 sampai 120, Menampilkan tombol Button bertuliskan "CALCULATE": dengan Lebar penuh (fillMaxWidth()). Tinggi 50dp, background berwarna sesuai tema utama. Dan ketika tombol ditekan, hasil perhitungan tipResult akan diisi dengan teks "Tip Amount: $" diikuti jumlah tip yang sudah diformat 2 angka desimal.
- Baris 122 samapi 127, Menampilkan teks hasil tip di bawah tombol, jika tipResult masih kosong, hanya akan tampil "Tip Amount". Dan teks ini diatur rata kanan (textAlign = TextAlign.End).
- Baris 133 smaapi 143, Fungsi TipOption() yaitu membuat satu baris Row yang berisi RadioButton dan teks keterangan, RadioButton akan otomatis aktif jika nilai selected sama dengan value. Dan saat dipilih, fungsi onSelect(value) akan mengubah nilai persentase tip (tipPercent).

MODUL 3 : Build a Scrollable List

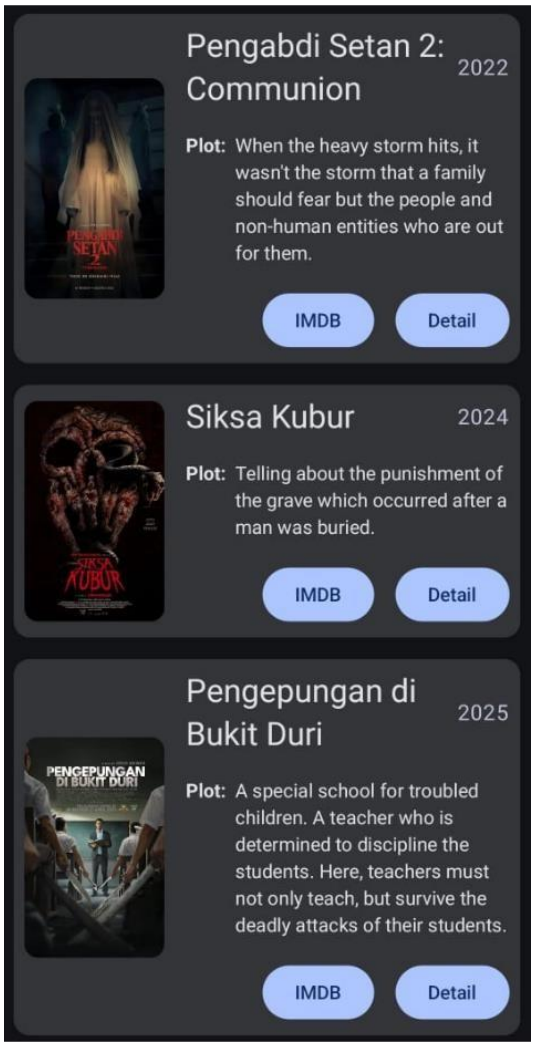
SOAL 1

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

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: Button
   a. pertama menggunakan intent eksplisit untuk membuka aplikasi atau browser lain
   b. Button kedua menggunakan Navigation component/intent untuk membuka laman detail item
3. Sudut item pada list dan gambar di dalam list melengkung atau rounded corner menggunakan Radius
4. 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
5. Aplikasi menggunakan arsitektur *single activity* (satu activity memiliki beberapa fragment)
6. 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. Soal 1 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. Soal 1 Modul 3

A. Source Code

Table 3. Sourcecode DinosaurRepositoryImpl Soal 1 Modul 3

- DinosaurRepositoryImpl.kt

```
1  package com.example.modul3.data.repository
2
3  import
4  com.example.modul3.data.datasource.LocalDinosaurDataSource
```

```
5  import com.example.modul3.domain.model.Dinosaur
6  import com.example.modul3.domain.repository.DinosaurRepository
7
8
9  class DinosaurRepositoryImpl : DinosaurRepository {
10     override fun getDinosaurs(): List<Dinosaur> {
11         return LocalDinosaurDataSource.getDinosaurs()
12     }
13 }
14
15
```

- LocalDinosaurDataSource

Table 4. Sourcecode localDinosaurDataSource Soal 1 Modul 3

```
1  package com.example.modul3.data.datasource
2
3  import com.example.modul3.R
4  import com.example.modul3.domain.model.Dinosaur
5
6  object LocalDinosaurDataSource {
7      fun getDinosaurs(): List<Dinosaur> {
8          return listOf(
9              Dinosaur(
10                 "Tyrannosaurus Rex",
11                 """
12                 Tyrannosaurus Rex adalah predator terbesar di
13 zamannya...
14                 """.trimIndent(),
15                 "Predator terbesar di zamannya dengan rahang
16 yang sangat kuat.",
17                 "Akhir Periode Kapur",
18                 R.drawable.tyrannosaurusrex,
19                 "Karnivora",
20                 "https://en.wikipedia.org/wiki/Tyrannosaurus"
```

```
            ),
            Dinosaur(
                "Triceratops",
                """
                Triceratops adalah dinosaurus herbivora besar...
                """.trimIndent(),
                "Herbivora besar dengan tiga tanduk ikonik di wajahnya.",
                "Akhir Periode Kapur",
                R.drawable.triceratops,
                "Herbivora",
                "https://en.wikipedia.org/wiki/Triceratops"
            ),
            // ... Lanjutkan untuk semua dinosaurus lainnya seperti kode di jawaban sebelumnya ...
            // Pastikan semua 10 dinosaurus memiliki data shortDesc dan period.
            // (Saya persingkat agar tidak terlalu panjang, gunakan kode lengkap dari jawaban sebelumnya)
            Dinosaur(
                "Velociraptor",
                """Velociraptor adalah dinosaurus kecil...""".trimIndent(),
                "Karnivora kecil yang terkenal karena kecepatan dan kecerdasannya.",
                "75-71 Juta Tahun Lalu",
                R.drawable.velociraptor, "Karnivora", "https://en.wikipedia.org/wiki/Velociraptor"
            ),
            Dinosaur(
                "Stegosaurus",
                """Stegosaurus adalah dinosaurus herbivora...""".trimIndent(),
```

```
                "Herbivora dengan piring tulang besar di
punggung dan ekor berduri.",
                "Periode Jurassic Akhir",
                R.drawable.stegosaurus, "Herbivora",
"https://en.wikipedia.org/wiki/Stegosaurus"
            ),
            Dinosaur(
                "Brachiosaurus",
                """Brachiosaurus adalah salah satu sauropoda
terbesar...""".trimIndent(),
                "Sauropoda raksasa dengan leher sangat panjang
dan kaki depan yang tinggi.",
                "Periode Jurassic",
                R.drawable.brachiosaurus, "Herbivora",
"https://en.wikipedia.org/wiki/Brachiosaurus"
            ),
            Dinosaur(
                "Spinosaurus",
                """Spinosaurus adalah dinosaurus karnivora
terbesar...""".trimIndent(),
                "Karnivora semi-akuatik dengan layar punggung
yang khas.",
                "112-93 Juta Tahun Lalu",
                R.drawable.spinosaurus, "Karnivora",
"https://en.wikipedia.org/wiki/Spinosaurus"
            ),
            Dinosaur(
                "Ankylosaurus",
                """Ankylosaurus adalah dinosaurus
herbivora...""".trimIndent(),
                "Herbivora berlapis zirah dengan ekor besar
berbentuk gada.",
                "Akhir Periode Kapur",
                R.drawable.ankylosaurus, "Herbivora",
```

```
"https://en.wikipedia.org/wiki/Ankylosaurus"
            ),
            Dinosaur(
                "Allosaurus",
                """Allosaurus adalah predator
utama...""".trimIndent(),
                "Predator puncak dari periode Jurassic, sering
disebut 'singa Jurassic'.",
                "Periode Jurassic Akhir",
                R.drawable.allosaurus, "Karnivora",
"https://en.wikipedia.org/wiki/Allosaurus"
            ),
            Dinosaur(
                "Diplodocus",
                """Diplodocus adalah sauropoda
raksasa...""".trimIndent(),
                "Salah satu dinosaurus terpanjang dengan leher
dan ekor super panjang.",
                "Periode Jurassic",
                R.drawable.diplodocus, "Herbivora",
"https://en.wikipedia.org/wiki/Diplodocus"
            ),
            Dinosaur(
                "Parasaurolophus",
                """Parasaurolophus adalah dinosaurus
herbivora...""".trimIndent(),
                "Herbivora dengan jambul panjang berongga yang
khas di kepalanya.",
                "Akhir Periode Kapur",
                R.drawable.parasaurolophus, "Herbivora",
"https://en.wikipedia.org/wiki/Parasaurolophus"
            )
        )
    }
```

```
117  }
118
119
```

- Dinosaur

Table 5. Sourcecode Dinosaur Soal 1 Modul 3

```
1  package com.example.modul3.domain.model
2
3  data class Dinosaur(
4      val name: String,
5      val description: String,
6      val shortDesc: String,
7      val period: String,
8      val imageRes: Int,
       val type: String,
       val wikiUrl: String
   )
```

- DinosaurRepository

Table 6. Sourcecode DinosaurRepository Soal 1 Modul 3

```
1  package com.example.modul3.domain.repository
2
3  import com.example.modul3.domain.model.Dinosaur
4
5  interface DinosaurRepository {
       fun getDinosaurs(): List<Dinosaur>
   }
```


- GetDinosauruseCase

Table 7. Sourcecode GetDinosaurCase Soal 1 Modul 3

```
1  package com.example.modul3.domain.usecase
2
3  import com.example.modul3.domain.model.Dinosaur
4  import com.example.modul3.domain.repository.DinosaurRepository
5
6  class GetDinosaursUseCase(private val repository:
7  DinosaurRepository) {
8      operator fun invoke(): List<Dinosaur> {
9          return repository.getDinosaurs()
10     }
11 }
```

- Appnavigation.kt

Table 8. Sourcecode AppNavigation Soal 1 Modul 3

```
1  package com.example.modul3.presentation.navigation
2
3  import androidx.compose.runtime.Composable
4  import androidx.compose.ui.Modifier
5  import androidx.navigation.NavHostController
6  import androidx.navigation.compose.NavHost
```

```
import androidx.navigation.compose.composable
import
com.example.modul3.presentation.dino_detail.DinoDetailScreen
import com.example.modul3.presentation.dino_list.DinoListScreen
import java.net.URLDecoder
import java.nio.charset.StandardCharsets

@Composable
fun AppNavigation(navController: NavHostController, modifier:
Modifier = Modifier) {
    NavHost(
        navController = navController,
        startDestination = "dino_list",
        modifier = modifier
    ) {
        composable("dino_list") {
            DinoListScreen(navController)
        }
        composable("detail/{name}/{desc}/{imgRes}") { backStack ->
            val name = backStack.arguments?.getString("name") ?:
""
            val encodedDesc =
backStack.arguments?.getString("desc") ?: ""
            val desc = URLDecoder.decode(encodedDesc,
StandardCharsets.UTF_8.toString())
            val imgRes =
backStack.arguments?.getString("imgRes")?.toIntOrNull()

            if (imgRes != null) {
                DinoDetailScreen(name, desc, imgRes,
navController)
            } else {

```

```
39              }
40          }
41      }
42  }
43
44
```

- DinoDetailScrenn.kt

Table 9. Sourcecode DinodetailScreen Soal 1 Modul 3

```
1   package com.example.modul3.presentation.dino_detail
2
3   import androidx.compose.foundation.Image
4   import androidx.compose.foundation.layout.*
5   import androidx.compose.foundation.rememberScrollState
6   import androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.foundation.verticalScroll
8   import androidx.compose.material3.Button
9   import androidx.compose.material3.Text
10  import androidx.compose.runtime.Composable
11  import androidx.compose.ui.Alignment
12  import androidx.compose.ui.Modifier
13  import androidx.compose.ui.draw.clip
14  import androidx.compose.ui.layout.ContentScale
15  import androidx.compose.ui.res.painterResource
16  import androidx.compose.ui.text.font.FontWeight
17  import androidx.compose.ui.text.style.TextAlign
18  import androidx.compose.ui.unit.dp
19  import androidx.compose.ui.unit.sp
20  import androidx.navigation.NavController
21
22  @Composable
23  fun DinoDetailScreen(
```

```
    name: String,
    desc: String,
    imgRes: Int,
    navController: NavController
) {
    Column(
        modifier = Modifier
            .padding(16.dp)
            .verticalScroll(rememberScrollState())
            .fillMaxSize()
    ) {
        Image(
            painter = painterResource(id = imgRes),
            contentDescription = null,
            modifier = Modifier
                .fillMaxWidth()
                .height(250.dp)
                .clip(RoundedCornerShape(20.dp)),
            contentScale = ContentScale.Crop
        )
        Spacer(modifier = Modifier.height(16.dp))
        Text(
            name,
            fontSize = 22.sp,
            fontWeight = FontWeight.Bold,
            modifier = Modifier.fillMaxWidth(),
            textAlign = TextAlign.Center
        )
        Spacer(modifier = Modifier.height(12.dp))
        Text(desc, fontSize = 14.sp, textAlign =
TextAlign.Justify)
        Spacer(modifier = Modifier.height(16.dp))
        Button(
            onClick = { navController.popBackStack() }, // Lebih
```

```
55  baik menggunakan popBackStack
56              modifier =
57  Modifier.align(Alignment.CenterHorizontally)
58          ) {
59              Text("Kembali")
60          }
61      }
62  }
```

- DinoListScreen.kt

Table 10. Sourcecode DinoListScreen Soal 1 Modul 3

```
1   package com.example.modul3.presentation.dino_list
2
3   import android.content.Intent
4   import android.net.Uri
5   import androidx.compose.foundation.Image
6   import androidx.compose.foundation.background
7   import androidx.compose.foundation.layout.*
8   import androidx.compose.foundation.lazy.grid.GridCells
9   import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
10  import androidx.compose.foundation.lazy.grid.items
11  import androidx.compose.foundation.shape.RoundedCornerShape
12  import androidx.compose.material3.Button
13  import androidx.compose.material3.ButtonDefaults
14  import androidx.compose.material3.Card
15  import androidx.compose.material3.CardDefaults
16  import androidx.compose.material3.DropdownMenu
17  import androidx.compose.material3.DropdownMenuItem
18  import androidx.compose.material3.OutlinedTextField
19  import androidx.compose.material3.Text
20  import androidx.compose.runtime.Composable
```

```
import androidx.compose.runtime.collectAsState
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextOverflow
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.lifecycle.viewmodel.compose.viewModel
import androidx.navigation.NavController
import java.net.URLEncoder
import java.nio.charset.StandardCharsets

@Composable
fun DinoListScreen(
    navController: NavController,
    vm: DinoListViewModel = viewModel()
) {
    val uiState by vm.uiState.collectAsState()
    val context = LocalContext.current
    val types = listOf("Semua", "Herbivora", "Karnivora")

    Column(
        modifier = Modifier
            .background(Color(0xFFF4F1DE))
            .fillMaxSize()
```

```
    ) {
        Box(
            modifier = Modifier
                .fillMaxWidth()
                .background(Color(0xFF81B29A))
                .padding(16.dp)
        ) {
            Text(
                text = "Jenis-Jenis Dinosaurus",
                fontSize = 24.sp,
                fontWeight = FontWeight.Bold,
                color = Color.White,
                modifier = Modifier.align(Alignment.CenterStart)
            )
        }

        OutlinedTextField(
            value = uiState.searchQuery,
            onValueChange = { vm.onSearchQueryChanged(it) },
            label = { Text("Cari dinosaurus...") },
            shape = RoundedCornerShape(24.dp),
            modifier = Modifier
                .fillMaxWidth()
                .padding(16.dp)
        )

        Row(
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp, vertical = 8.dp),
            horizontalArrangement = Arrangement.SpaceBetween,
            verticalAlignment = Alignment.CenterVertically
        ) {
            Text("Filter berdasarkan tipe:", fontSize = 14.sp)
```

```
84              FilterDropdownMenu(
85                  options = types,
86                  selectedOption = uiState.selectedFilter,
87                  onOptionSelected = { vm.onFilterChanged(it) }
88              )
89          }
90
91          LazyVerticalGrid(
92              columns = GridCells.Fixed(2),
93              contentPadding = PaddingValues(16.dp),
94              verticalArrangement = Arrangement.spacedBy(16.dp),
95              horizontalArrangement = Arrangement.spacedBy(16.dp),
96          ) {
97              items(uiState.filteredDinosaurs) { dino ->
98                  Card(
99                      modifier = Modifier.fillMaxWidth(),
100                     colors =
CardDefaults.cardColors(containerColor = Color(0xFFFFF7F0)),
101                     elevation =
102 CardDefaults.cardElevation(defaultElevation = 4.dp),
103                     shape = RoundedCornerShape(16.dp)
104                 ) {
105                     Column {
106                         Image(
107                             painter = painterResource(id =
108 dino.imageRes),
109                             contentDescription = dino.name,
110                             contentScale = ContentScale.Crop,
111                             modifier = Modifier
112                                 .fillMaxWidth()
113                                 .height(120.dp)
114                                 .clip(RoundedCornerShape(topStart
115 = 16.dp, topEnd = 16.dp))
                        )
```

```
                        Column(modifier =
Modifier.padding(12.dp)) {
                            Text(
                                text = dino.name,
                                fontSize = 16.sp,
                                fontWeight = FontWeight.Bold,
                                color = Color(0xFF3D405B),
                                maxLines = 1,
                                overflow = TextOverflow.Ellipsis
                            )
                            Text(
                                text = dino.period,
                                fontSize = 12.sp,
                                color = Color.Gray
                            )
                            Spacer(modifier =
Modifier.height(8.dp))

                            Text(
                                text = dino.shortDesc,
                                fontSize = 13.sp,
                                color = Color.DarkGray,
                                maxLines = 3,
                                overflow = TextOverflow.Ellipsis,
                                modifier = Modifier.heightIn(min
= 50.dp)
                            )
                            Spacer(modifier =
Modifier.height(12.dp))

                            Row(
                                modifier =
Modifier.fillMaxWidth(),
                                horizontalArrangement =
```

```
148 Arrangement.SpaceBetween
149                             ) {
150                                 Button(
151                                     onClick = {
152                                         val encodedDesc =
153 URLEncoder.encode(dino.description,
154 StandardCharsets.UTF_8.toString())
155
156 navController.navigate("detail/${dino.name}/$encodedDesc/${dino.i
157 mageRes}")
158                                     },
159                                     colors =
160 ButtonDefaults.buttonColors(containerColor = Color(0xFF81B29A)),
161                                     shape =
162 RoundedCornerShape(12.dp),
163                                     modifier =
164 Modifier.weight(1f),
165                                     contentPadding =
166 PaddingValues(horizontal = 4.dp, vertical = 8.dp)
167                                 ) {
168                                     Text("Info Detail", color =
169 Color.White, fontSize = 12.sp)
170                                 }
171                                 Spacer(modifier =
172 Modifier.width(8.dp))
173                                 Button(
174                                     onClick = {
175                                         val intent =
176 Intent(Intent.ACTION_VIEW, Uri.parse(dino.wikiUrl))
177
178 context.startActivity(intent)
179                                     },
                                    colors =
ButtonDefaults.buttonColors(containerColor = Color(0xFFE07A5F)),
```

```
                                    shape =
RoundedCornerShape(12.dp),
                                    modifier =
Modifier.weight(1f),
                                    contentPadding =
PaddingValues(horizontal = 4.dp, vertical = 8.dp)
                                ) {
                                    Text("Wikipedia", color =
Color.White, fontSize = 12.sp)
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}


@Composable
fun FilterDropdownMenu(
    options: List<String>,
    selectedOption: String,
    onOptionSelected: (String) -> Unit
) {
    var expanded by remember { mutableStateOf(false) }
    Box {
        Button(
            onClick = { expanded = true },
            colors = ButtonDefaults.buttonColors(containerColor =
Color(0xFF81B29A))
        ) {
            Text(selectedOption, color = Color.White)
```

```
212         }
213         DropdownMenu(
214             expanded = expanded,
215             onDismissRequest = { expanded = false }
216         ) {
217             options.forEach { label ->
218                 DropdownMenuItem(
219                     text = { Text(label) },
220                     onClick = {
221                         onOptionSelected(label)
222                         expanded = false
223                     }
224                 )
225             }
226         }
227     }
228
```

- DinoListviewmodel.kt

Table 11. Sourcecode DinoListViewModel Soal 1 Modul 3

```
1   package com.example.modul3.presentation.dino_list
2
3   import androidx.lifecycle.ViewModel
4   import com.example.modul3.data.repository.DinosaurRepositoryImpl
5   import com.example.modul3.domain.model.Dinosaur
6   import com.example.modul3.domain.usecase.GetDinosaursUseCase
7   import kotlinx.coroutines.flow.MutableStateFlow
8   import kotlinx.coroutines.flow.StateFlow
9   import kotlinx.coroutines.flow.asStateFlow
10  import kotlinx.coroutines.flow.update
```

```
11
12
13  data class DinoListUiState(
14      val allDinosaurs: List<Dinosaur> = emptyList(),
15      val filteredDinosaurs: List<Dinosaur> = emptyList(),
16      val searchQuery: String = "",
17      val selectedFilter: String = "Semua"
18  )
19
20
21  class DinoListViewModel : ViewModel() {
22
23      private val getDinosaursUseCase =
    GetDinosaursUseCase(DinosaurRepositoryImpl())
24
25      private val _uiState = MutableStateFlow(DinoListUiState())
26      val uiState: StateFlow<DinoListUiState> =
27  _uiState.asStateFlow()
28
29      init {
30          loadDinosaurs()
31      }
32
33      private fun loadDinosaurs() {
34          val dinosaurs = getDinosaursUseCase()
35          _uiState.update {
36              it.copy(
37                  allDinosaurs = dinosaurs,
38                  filteredDinosaurs = dinosaurs
39              )
40          }
41      }
42
      fun onSearchQueryChanged(query: String) {
```

```
43          _uiState.update { it.copy(searchQuery = query) }
44          filterDinosaurs()
45      }
46
47      fun onFilterChanged(filter: String) {
48          _uiState.update { it.copy(selectedFilter = filter) }
49          filterDinosaurs()
50      }
51
52      private fun filterDinosaurs() {
53          val currentState = _uiState.value
54          val filteredList = currentState.allDinosaurs.filter {
dinosaur ->
55              val matchesSearch =
56  dinosaur.name.contains(currentState.searchQuery, ignoreCase =
57  true)
58              val matchesFilter = currentState.selectedFilter ==
59  "Semua" || dinosaur.type == currentState.selectedFilter
60              matchesSearch && matchesFilter
61          }
62          _uiState.update { it.copy(filteredDinosaurs =
63  filteredList) }
64      }
    }
```

- MainActivity.kt

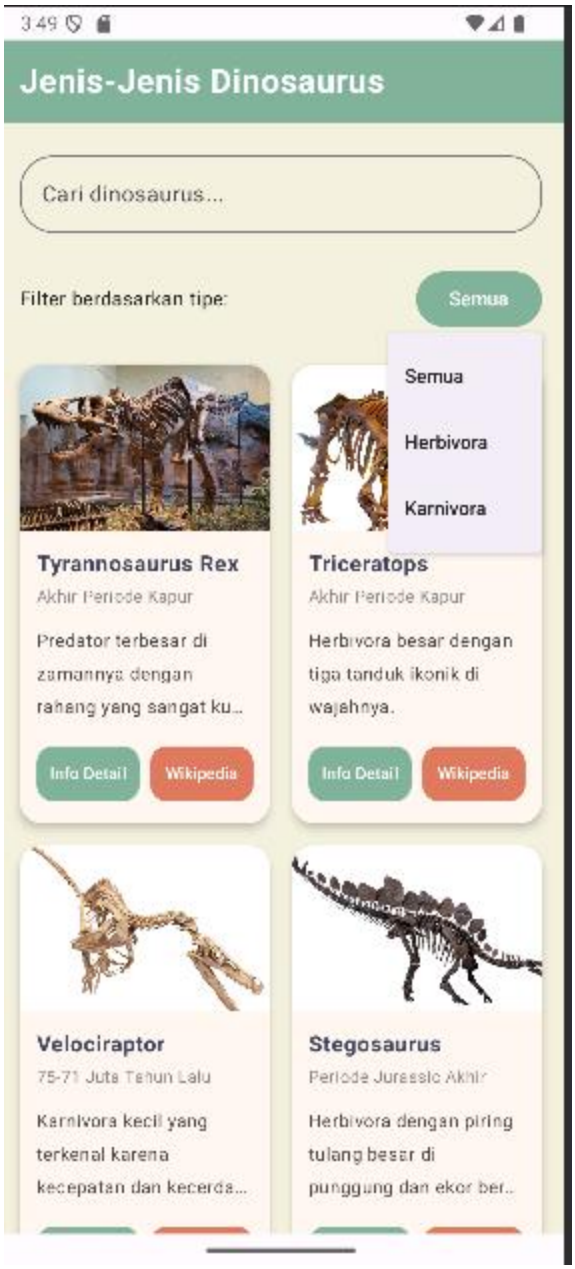
Table 12. Sourcecode MainActivity Soal 1 Modul 3

```
1   package com.example.modul3.presentation
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.padding
8  import androidx.compose.material3.*
9  import androidx.compose.ui.Modifier
10 import androidx.navigation.compose.rememberNavController
11 import com.example.modul3.presentation.navigation.AppNavigation
12
13 class MainActivity : ComponentActivity() {
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         enableEdgeToEdge()
17         setContent {
18             MaterialTheme {
19                 val navController = rememberNavController()
20                 Scaffold { padding ->
21                     AppNavigation(
22                         navController = navController,
23                         modifier = Modifier.padding(padding)
24                     )
25                 }
26             }
27     }
28 }
29
30
```

B. Output Program

Gambar 13. Screenshot Hasil Jawaban Soal 1 Modul 3



Gambar 14. Screenshot Hasil Jawaban Soal 1 Modul 3

C. Pembahasan

- Dinosaur.kt:

File ini menginisiasi sebuah data class bernama **Dinosaur**. Kelas ini berfungsi sebagai model data atau cetak biru untuk objek dinosaurus di dalam aplikasi. Setiap objek Dinosaur akan memiliki lima properti: name (nama dinosaurus), description (deskripsi panjang), imageRes (ID resource gambar dari drawable), type (jenisnya, seperti Karnivora atau Herbivora), dan wikiUrl (URL ke halaman Wikipedia-nya) dan lain lain

- LocalDinosaurDataSource.kt

File ini berfungsi sebagai sumber data lokal dan statis untuk aplikasi. Di dalamnya terdapat satu fungsi, **getDinosaurs()**, yang membuat dan mengembalikan sebuah List dari objek Dinosaur. Semua data dinosaurus, termasuk nama, deskripsi, gambar, dan URL, didefinisikan secara *hardcoded* di sini. Anggapanya adalah database dari aplikasi dinosaurus.

- DinosaurusRepository.kt

File ini berfungsi sebagai interface, lalu mendeklarasikan satu fungsi abstrak, getDinosaurs(), yang mewajibkan setiap kelas yang mengimplementasikannya untuk menyediakan cara mengembalikan sebuah List<Dinosaur>. Tujuan utama dari interface ini adalah untuk menciptakan abstraksi, memisahkan logika bisnis dari cara spesifik pengambilan data, sehingga di masa depan sumber data dapat diganti (misalnya, dari lokal ke internet) tanpa mengubah bagian lain dari aplikasi.

- DinosaurRepositoryImpl.kt

File ini berisi implementasi konkret dari DinosaurRepository. kelas ini menggunakan kata kunci override untuk menyediakan implementasi fungsi getDinosaurs(), lalu file ini memiliki cara dengan memanggil metode LocalDinosaurDataSource.getDinosaurs() dan langsun mengembalikan hasilnya.

- GetDinosaurUseCase.kt

File ini berguna untuk mendapatkan daftar dinosaurus, lalu class ini bergantung kepada DinosaurRepository yang diterimanya melalui constructor.

- DinoListViewModel.kt

Kelas ini menggunakan StateFlow untuk menyimpan dan mengekspos DinoListUiState, sebuah data class yang menampung semua status layar seperti daftar lengkap dinosaurus, daftar yang telah difilter, query pencarian, dan jenis filter. Saat ViewModel pertama kali dibuat, dan class ini langsung memuat data dinosaurus melalui GetDinosaursUseCase.

- DinoListScreen.kt

File ini berisi Composable yang bertanggung jawab untuk membangun antarmuka pengguna (UI) dari daftar dinosaurus. Fungsi ini mengamati uiState dari DinoListViewModel menggunakan collectAsState, sehingga UI akan otomatis diperbarui setiap kali ada perubahan data. Lalu disini menggunakan LazyVerticalGrid untuk menampilkan item secara efisien dalam format dua kolom. Setiap item di dalam grid adalah sebuah Card yang berisi gambar, teks, serta dua tombol dengan fungsi berbeda: satu tombol memicu navigasi ke DinoDetailScreen melalui navController, dan satu lagi memicu Intent untuk membuka browser ke halaman Wikipedia.

- DinoDetailScreen.kt

File ini bertujuan untuk menampilkan halaman detail dari satu dinosaurus yang dipilih. Fungsi ini menerima data seperti nama, deskripsi, dan ID gambar sebagai parameter yang dilewatkan melalui proses navigasi. Tampilan UI-nya disusun secara vertikal menggunakan Column, menampilkan Image dari dinosaurus, diikuti oleh Text untuk nama dan deskripsi lengkap. Dan juga terdapat button kembali yang memanggil navController.popBackStack()

- AppNavigation.kt

File ini bertujuan untuk mengatur navigasi dalam aplikasi menggunakan Jetpack Navigation Compose. Di dalamnya, Composable AppNavigation mendefinisikan sebuah NavHost yang menjadi wadah untuk semua rute atau tujuan navigasi. startDestination diatur ke "dino_list" sebagai layar awal. Terdapat dua rute utama: "dino_list" untuk DinoListScreen, dan "detail/{name}/{desc}/{imgRes}" untuk DinoDetailScreen, di mana rute kedua ini menggunakan placeholder ({...}) untuk menerima argumen.

- MainActivity,kt

File ini adalah titik awal pada aplikasi ini, Di dalam fungsi onCreate, ia menggunakan setContent untuk membangun seluruh antarmuka pengguna dengan Jetpack Compose. Pafa bagian ini menginisialisasi rememberNavController() untuk membuat dan mengingat NavController yang akan mengelola semua perpindahan layar. Struktur dasar aplikasi diatur menggunakan Scaffold, dan komponen AppNavigation dipanggil di dalamnya.

2. **RecyclerView** masih banyak digunakan karena dua alasan utama: untuk **proyek yang sudah ada (legacy)** yang dibangun dengan sistem XML, dan ketika developer membutuhkan **kustomisasi tingkat lanjut** pada perilaku list yang belum tentu mudah dilakukan di **LazyColumn**.

   Penjelasan Detail:

   - Sebagian besar aplikasi Android yang ada saat ini dibangun menggunakan sistem **View berbasis XML**, bukan Compose.
   - **RecyclerView** memberikan kontrol penuh atas animasi item (menambah, menghapus, memindahkan) melalui RecyclerView.ItemAnimator. Developer bisa membuat animasi yang sangat kompleks dan spesifik.

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

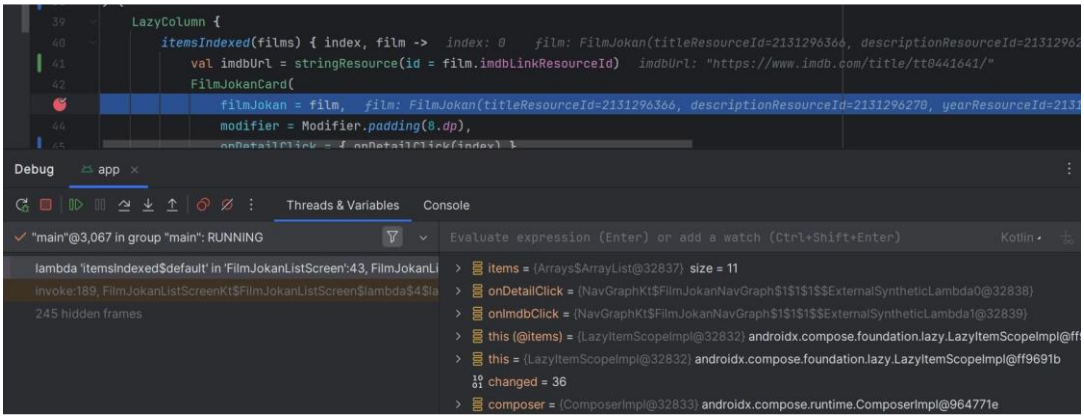
   b. Log saat tombol Detail dan tombol Explicit Intent ditekan

   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

- DinoRepositoryImpl.kt

Table 13. Sourcecode DinoRepositoryImpl Soal 1 Modul 4

```
1   package com.example.modul4.data.repository
2
3   import com.example.modul4.R
4   import com.example.modul4.domain.model.Dino
5   import com.example.modul4.domain.repository.DinoRepository
6   import kotlinx.coroutines.flow.Flow
7   import kotlinx.coroutines.flow.flowOf
8
9   class DinoRepositoryImpl : DinoRepository {
10
11      private val dinos = listOf(
12          Dino(
13              name = "Tyrannosaurus Rex",
14              description = """
15                  Tyrannosaurus Rex adalah predator terbesar di
16  zamannya, hidup sekitar 68 hingga 66 juta tahun yang lalu di
17  akhir periode Kapur. Dengan panjang mencapai 12 meter dan
18  tinggi hampir 4 meter, T-Rex memiliki rahang sangat kuat dan
19  gigi setajam pisau untuk merobek daging mangsanya.
20
21                  Meskipun tangan depannya kecil, tubuhnya yang
22  besar dan kekuatan gigitan luar biasa menjadikannya salah satu
23  dinosaurus paling menakutkan. Fosilnya ditemukan di Amerika
24  Utara dan menjadi ikon dalam dunia paleontologi.
25              """.trimIndent(),
26              shortDesc = "Predator terbesar di zamannya dengan
27  rahang yang sangat kuat.",
28              period = "Akhir Periode Kapur",
29              imageRes = R.drawable.tyrannosaurusrex,
30              type = "Karnivora",
31              wikiUrl =
32
33
```

```
"https://en.wikipedia.org/wiki/Tyrannosaurus"
        ),
        Dino(
            name = "Triceratops",
            description = """
                Triceratops adalah dinosaurus herbivora besar dengan tiga tanduk dan pelindung tengkorak besar, hidup berdampingan dengan T-Rex di akhir Kapur sekitar 68 juta tahun yang lalu. Ia mencapai panjang sekitar 9 meter dan berat hingga 12 ton.

                Tanduk panjangnya diyakini digunakan untuk pertahanan diri dan pertarungan sesama jantan saat musim kawin. Struktur tengkoraknya juga mungkin berfungsi untuk pengaturan suhu tubuh atau sebagai alat komunikasi visual.
            """.trimIndent(),
            shortDesc = "Herbivora besar dengan tiga tanduk ikonik di wajahnya.",
            period = "Akhir Periode Kapur",
            imageRes = R.drawable.triceratops,
            type = "Herbivora",
            wikiUrl = "https://en.wikipedia.org/wiki/Triceratops"
        ),
        Dino(
            name = "Velociraptor",
            description = """
                Velociraptor adalah dinosaurus kecil dan karnivora yang terkenal karena kecepatannya dan diduga memiliki bulu. Ia hidup sekitar 75 hingga 71 juta tahun yang lalu di wilayah yang kini merupakan Mongolia.

                Meski hanya seukuran kalkun, Velociraptor
```

```
sangat gesit dan cerdas, serta berburu dalam kelompok. Cakar
melengkung pada kaki belakangnya digunakan untuk mencengkram
dan melumpuhkan mangsa.
            """.trimIndent(),
            shortDesc = "Karnivora kecil yang terkenal karena
kecepatan dan kecerdasannya.",
            period = "75-71 Juta Tahun Lalu",
            imageRes = R.drawable.velociraptor,
            type = "Karnivora",
            wikiUrl =
"https://en.wikipedia.org/wiki/Velociraptor"
        ),
        Dino(
            name = "Stegosaurus",
            description = """
                Stegosaurus adalah dinosaurus herbivora dari
periode Jurassic akhir, terkenal dengan piring tulang besar di
punggung dan ekor berduri yang disebut thagomizer. Ia hidup
sekitar 155 juta tahun lalu dan memiliki otak relatif kecil.

                Piring punggungnya kemungkinan digunakan untuk
menakuti pemangsa, menarik pasangan, atau mengatur suhu tubuh.
Ekor berdurinya menjadi senjata efektif melawan predator
seperti Allosaurus.
            """.trimIndent(),
            shortDesc = "Herbivora dengan piring tulang besar
di punggung dan ekor berduri.",
            period = "Periode Jurassic Akhir",
            imageRes = R.drawable.stegosaurus,
            type = "Herbivora",
            wikiUrl =
```

```
"https://en.wikipedia.org/wiki/Stegosaurus"
        ),
        Dino(
            name = "Brachiosaurus",
            description = """
                Brachiosaurus adalah salah satu sauropoda terbesar yang hidup sekitar 154 hingga 150 juta tahun lalu pada periode Jurassic. Ciri khasnya adalah leher panjang yang memungkinkan ia meraih daun dari pohon tinggi.

                Berbeda dari sauropoda lain, kaki depan Brachiosaurus lebih panjang daripada kaki belakang, membuat posturnya menjulang tinggi. Ia diperkirakan memiliki berat hingga 40 ton dan panjang lebih dari 25 meter.
            """.trimIndent(),
            shortDesc = "Sauropoda raksasa dengan leher sangat panjang dan kaki depan yang tinggi.",
            period = "Periode Jurassic",
            imageRes = R.drawable.brachiosaurus,
            type = "Herbivora",
            wikiUrl =
"https://en.wikipedia.org/wiki/Brachiosaurus"
        ),
        Dino(
            name = "Spinosaurus",
            description = """
                Spinosaurus adalah dinosaurus karnivora terbesar yang hidup sekitar 112–93 juta tahun lalu di Afrika Utara. Ciri utamanya adalah layar punggung tinggi seperti layar kapal, kemungkinan digunakan untuk menampilkan diri atau mengatur suhu.

                Ia diyakini sebagai semi-akuatik, berburu ikan
```

```
dan hewan air lainnya di sungai purba. Dengan panjang mencapai
15 meter, Spinosaurus lebih besar dari T-Rex dan sangat adaptif
terhadap lingkungan air.
            """.trimIndent(),
            shortDesc = "Karnivora semi-akuatik dengan layar
punggung yang khas.",
            period = "112-93 Juta Tahun Lalu",
            imageRes = R.drawable.spinosaurus,
            type = "Karnivora",
            wikiUrl =
"https://en.wikipedia.org/wiki/Spinosaurus"
        ),
        Dino(
            name = "Ankylosaurus",
            description = """
                Ankylosaurus adalah dinosaurus herbivora dari
akhir periode Kapur, terkenal karena tubuhnya yang dilapisi
zira dan ekor besar berbentuk gada. Panjangnya mencapai 6
hingga 8 meter dan berat hingga 8 ton.

                Bentuk tubuhnya seperti tank membuatnya hampir
kebal terhadap serangan predator. Ekor berotot dan keras dapat
digunakan untuk menyerang balik dan melukai lawan secara fatal.
            """.trimIndent(),
            shortDesc = "Herbivora berlapis zirah dengan ekor
besar berbentuk gada.",
            period = "Akhir Periode Kapur",
            imageRes = R.drawable.ankylosaurus,
            type = "Herbivora",
            wikiUrl =
```

```
"https://en.wikipedia.org/wiki/Ankylosaurus"
        ),
        Dino(
            name = "Allosaurus",
            description = """
                Allosaurus adalah predator utama dari periode Jurassic, hidup sekitar 155 hingga 145 juta tahun lalu. Ia memiliki tengkorak besar, gigi tajam, dan tubuh ramping yang memungkinkannya berburu dengan kecepatan tinggi.

                Allosaurus sering disebut 'singa Jurassic' karena perannya sebagai pemburu puncak. Ia mungkin berburu dalam kelompok dan mangsanya termasuk sauropoda besar seperti Diplodocus dan Camarasaurus.
            """.trimIndent(),
            shortDesc = "Predator puncak dari periode Jurassic, sering disebut 'singa Jurassic'.",
            period = "Periode Jurassic Akhir",
            imageRes = R.drawable.allosaurus,
            type = "Karnivora",
            wikiUrl =
"https://en.wikipedia.org/wiki/Allosaurus"
        ),
        Dino(
            name = "Diplodocus",
            description = """
                Diplodocus adalah sauropoda raksasa dari periode Jurassic, dikenal karena leher dan ekor super panjang. Panjang tubuhnya bisa mencapai 27 meter, menjadikannya salah satu dinosaurus terpanjang yang pernah hidup.

                Ia hidup di Amerika Utara sekitar 154 juta tahun lalu dan memakan tumbuhan rendah di hutan purba. Ekor
```

```
panjangnya kemungkinan digunakan untuk pertahanan atau
komunikasi sonik seperti cambuk.
            """.trimIndent(),
            shortDesc = "Salah satu dinosaurus terpanjang
dengan leher dan ekor super panjang.",
            period = "Periode Jurassic",
            imageRes = R.drawable.diplodocus,
            type = "Herbivora",
            wikiUrl =
"https://en.wikipedia.org/wiki/Diplodocus"
        ),
        Dino(
            name = "Parasaurolophus",
            description = """
                Parasaurolophus adalah dinosaurus herbivora
dari akhir Kapur yang dikenal karena jambul panjang berongga di
kepalanya. Jambul tersebut kemungkinan digunakan untuk
menghasilkan suara, menarik pasangan, atau membantu mengatur
suhu tubuh.

                Ia berjalan dengan dua atau empat kaki dan
hidup dalam kawanan. Panjang tubuhnya sekitar 10 meter dan
dikenal sebagai bagian dari kelompok hadrosaur atau dinosaurus
bebek.
            """.trimIndent(),
            shortDesc = "Herbivora dengan jambul panjang
berongga yang khas di kepalanya.",
            period = "Akhir Periode Kapur",
            imageRes = R.drawable.parasaurolophus,
            type = "Herbivora",
            wikiUrl =


```

```
227 "https://en.wikipedia.org/wiki/Parasaurolophus"
228         )
229     )
230
231     fun getDinoList(): List<Dino> = dinos
232
233     fun getDinoByName(name: String): Dino? =
234         dinos.find { it.name.equals(name, ignoreCase = true) }
235
236     fun getDinoFlow(): Flow<List<Dino>> = flowOf(dinos)
237
238     override fun getAllDinos(): Flow<List<Dino>> =
239 flowOf(dinos)
240
241     override fun searchDinos(query: String): Flow<List<Dino>> =
242         flowOf(dinos.filter { it.name.contains(query,
    ignoreCase = true) })
```

- Dino.kt

Table 14. Sourcecode Dino Soal 1 Modul 4

```
1  package com.example.modul4.domain.model
2
3  data class Dino(
4      val name: String,
5      val description: String,
6      val shortDesc: String,
7      val period: String,
8      val imageRes: Int,
9      val type: String,
10     val wikiUrl: String
11 )
12
```

- DinoRepository.kt

Table 15. Sourcecode DinoRepository Soal 1 Modul 4

```
1  package com.example.modul4.domain.repository
2
3  import com.example.modul4.domain.model.Dino
4  import kotlinx.coroutines.flow.Flow
5
6  interface DinoRepository {
7      fun getAllDinos(): Flow<List<Dino>>
8      fun searchDinos(query: String): Flow<List<Dino>>
9  }
```

- GetDinoListUseCase.kt

Table 16. Sourcecode GetDinoListUseCase Soal 1 Modul 4

```
1  package com.example.modul4.domain.usecase
2
3  import com.example.modul4.domain.model.Dino
4  import com.example.modul4.domain.repository.DinoRepository
5  import kotlinx.coroutines.flow.Flow
6
7  class SearchDinosUseCase(private val repository: DinoRepository)
8  {
9      operator fun invoke(query: String): Flow<List<Dino>> =
10         repository.searchDinos(query)
11 }
12 }
13
14
```

- DinoApp.kt

Table 17. Sourcecode DinoApp Soal 1 Modul 4

```
1  package com.example.modul4.ui.theme.presentation
2
3  import androidx.compose.runtime.Composable
4  import androidx.lifecycle.viewmodel.compose.viewModel
5  import androidx.navigation.NavHostController
6  import androidx.navigation.compose.NavHost
7  import androidx.navigation.compose.composable
8  import androidx.navigation.compose.rememberNavController
9  import
10 com.example.modul4.presentation.ui.screen.DinoDetailScreen
11 import com.example.modul4.presentation.ui.screen.DinoListScreen
12 import com.example.modul4.presentation.viewmodel.DinoViewModel
13 import
```

```
14  com.example.modul4.presentation.viewmodel.DinoViewModelFactory
15
16  @Composable
17  fun DinoApp() {
18
19      val navController: NavHostController =
20  rememberNavController()
21
22      val viewModel: DinoViewModel = viewModel(factory =
24  DinoViewModelFactory())
24
25
26      NavHost(navController = navController, startDestination =
27  "dino_list") {
28          composable("dino_list") {
29              DinoListScreen(navController = navController,
30  viewModel = viewModel)
31          }
32
33          composable("detail") {
34              DinoDetailScreen(viewModel = viewModel,
35  navController = navController)
36          }
37      }
38  }
39
40
41
42
43
44
```

- DinoDetailScreen.kt

Table 18. Sourcecode dinoDetailScreen Soal 1 Modul 4

```
1  package com.example.modul4.presentation.ui.screen
2
3  import androidx.compose.foundation.Image
4  import androidx.compose.foundation.background
5  import androidx.compose.foundation.layout.*
6  import androidx.compose.foundation.rememberScrollState
7  import androidx.compose.foundation.shape.RoundedCornerShape
8  import androidx.compose.foundation.verticalScroll
9  import androidx.compose.material3.Button
10 import androidx.compose.material3.ButtonDefaults
11 import androidx.compose.material3.Text
12 import androidx.compose.runtime.Composable
13 import androidx.compose.runtime.collectAsState
14 import androidx.compose.runtime.getValue
15 import androidx.compose.ui.Alignment
16 import androidx.compose.ui.Modifier
17 import androidx.compose.ui.draw.clip
18 import androidx.compose.ui.graphics.Color
19 import androidx.compose.ui.res.painterResource
20 import androidx.compose.ui.text.font.FontWeight
21 import androidx.compose.ui.text.style.TextAlign
22 import androidx.compose.ui.unit.dp
24 import androidx.compose.ui.unit.sp
24 import androidx.navigation.NavController
25 import com.example.modul4.presentation.viewmodel.DinoViewModel
26
27 @Composable
28 fun DinoDetailScreen(
29     viewModel: DinoViewModel,
30     navController: NavController
31 ) {
32     val selectedDino by viewModel.selectedDino.collectAsState()
33
```

```
34      if (selectedDino == null) {
35          Box(modifier = Modifier.fillMaxSize(), contentAlignment
36  = Alignment.Center) {
37              Text("Data tidak tersedia")
38          }
39          return
40      }
41
42      val dino = selectedDino!!
43
44      Column(
45          modifier = Modifier
46              .fillMaxSize()
47              .background(Color(0xFFF4F1DE))
48              .verticalScroll(rememberScrollState())
49              .padding(16.dp)
50      ) {
51          Image(
52              painter = painterResource(id = dino.imageRes),
53              contentDescription = dino.name,
54              modifier = Modifier
55                  .fillMaxWidth()
56                  .height(250.dp)
57                  .clip(RoundedCornerShape(20.dp))
58          )
59          Spacer(modifier = Modifier.height(16.dp))
60          Text(
61              text = dino.name,
62              fontSize = 24.sp,
63              fontWeight = FontWeight.Bold,
64              textAlign = TextAlign.Center,
65              color = Color(0xFF3D405B),
66              modifier = Modifier.fillMaxWidth()
67          )
```

```
68          Spacer(modifier = Modifier.height(12.dp))
69          Text(
70              text = dino.description,
71              fontSize = 14.sp,
72              textAlign = TextAlign.Justify,
73              lineHeight = 20.sp,
74              color = Color(0xFF3D405B)
75          )
76          Spacer(modifier = Modifier.height(24.dp))
77          Button(
78              onClick = { navController.popBackStack() },
79              modifier =
80  Modifier.align(Alignment.CenterHorizontally),
81              colors = ButtonDefaults.buttonColors(containerColor
82  = Color(0xFFE07A5F))
83          ) {
84              Text("Kembali", color = Color.White)
85          }
86          Spacer(modifier = Modifier.height(16.dp))
87      }
88  }
89
```

- DinoListScreen.kt

Table 19. Sourcecode DinoListScreen Soal 1 Modul 4

```
1  package com.example.modul4.presentation.ui.screen
2
3  import android.content.Intent
4  import android.net.Uri
5  import android.util.Log
6  import androidx.compose.foundation.Image
7  import androidx.compose.foundation.background
8  import androidx.compose.foundation.layout.*
9  import androidx.compose.foundation.lazy.grid.GridCells
10 import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
11 import androidx.compose.foundation.lazy.grid.items
12 import androidx.compose.foundation.shape.RoundedCornerShape
13 import androidx.compose.material3.*
14 import androidx.compose.runtime.*
15 import androidx.compose.ui.Alignment
16 import androidx.compose.ui.Modifier
17 import androidx.compose.ui.draw.clip
18 import androidx.compose.ui.graphics.Color
19 import androidx.compose.ui.layout.ContentScale
20 import androidx.compose.ui.platform.LocalContext
21 import androidx.compose.ui.res.painterResource
22 import androidx.compose.ui.text.font.FontWeight
23 import androidx.compose.ui.unit.dp
24 import androidx.compose.ui.unit.sp
25 import androidx.navigation.NavController
26 import com.example.modul4.presentation.viewmodel.DinoViewModel
27
28 @Composable
29 fun DinoListScreen(
30     navController: NavController,
31     viewModel: DinoViewModel
```

```
) {
    val context = LocalContext.current
    var searchQuery by remember { mutableStateOf("") }
    var selectedFilter by remember { mutableStateOf("Semua") }
    val types = listOf("Semua", "Herbivora", "Karnivora")
    val dinoList by viewModel.dinoList.collectAsState()

    val filteredList = dinoList.filter {
        it.name.contains(searchQuery, ignoreCase = true) &&
                (selectedFilter == "Semua" || it.type ==
selectedFilter)
    }

    Column(
        modifier = Modifier
            .background(Color(0xFFF4F1DE))
            .fillMaxSize()
    ) {
        Box(
            modifier = Modifier
                .fillMaxWidth()
                .background(Color(0xFF81B29A))
                .padding(16.dp)
        ) {
            Text(
                text = "Jenis-Jenis Dinosaurus",
                fontSize = 24.sp,
                fontWeight = FontWeight.Bold,
                color = Color.White,
                modifier = Modifier.align(Alignment.CenterStart)
            )
        }


```

```
66          OutlinedTextField(
67              value = searchQuery,
68              onValueChange = { searchQuery = it },
69              label = { Text("Cari dinosaurus...") },
70              shape = RoundedCornerShape(24.dp),
71              modifier = Modifier
72                  .fillMaxWidth()
73                  .padding(16.dp)
74          )
75
76
77          Row(
78              modifier = Modifier
79                  .fillMaxWidth()
80                  .padding(horizontal = 16.dp),
81              horizontalArrangement = Arrangement.SpaceBetween,
82              verticalAlignment = Alignment.CenterVertically
83          ) {
84              Text("Filter berdasarkan tipe:", fontSize = 14.sp,
85  color = Color(0xFF3D405B))
86              DropdownMenuBox(types, selectedFilter) {
87  selectedFilter = it }
88          }
89
90          LazyVerticalGrid(
91              columns = GridCells.Fixed(2),
92              contentPadding = PaddingValues(16.dp),
93              verticalArrangement = Arrangement.spacedBy(12.dp),
94              horizontalArrangement = Arrangement.spacedBy(12.dp),
95              modifier = Modifier.fillMaxSize()
96          ) {
97              items(filteredList) { dino ->
98                  Card(
99                      modifier = Modifier.fillMaxWidth(),
```

```
                    colors =
CardDefaults.cardColors(containerColor = Color(0xFFFFF7F0)),
                    elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp),
                    shape = RoundedCornerShape(16.dp)
                ) {
                    Column {
                        Image(
                            painter = painterResource(id =
dino.imageRes),
                            contentDescription = dino.name,
                            contentScale = ContentScale.Crop,
                            modifier = Modifier
                                .fillMaxWidth()
                                .height(140.dp)
                                .clip(RoundedCornerShape(topStart
= 16.dp, topEnd = 16.dp))
                        )
                        Column(modifier = Modifier.padding(8.dp))
{
                            Spacer(modifier =
Modifier.height(6.dp))
                            Text(
                                dino.name,
                                fontSize = 16.sp,
                                fontWeight = FontWeight.Bold,
                                color = Color(0xFF3D405B)
                            )
                            Spacer(modifier =
Modifier.height(6.dp))

                            Row(
                                modifier =
Modifier.fillMaxWidth(),
```

```
                                horizontalArrangement =
Arrangement.SpaceBetween
                            ) {
                                Button(
                                    onClick = {
                                        Log.d("DinoListScreen",
"Tombol Detail ditekan untuk: ${dino.name}")

viewModel.selectDino(dino)

navController.navigate("detail")
                                    },
                                    colors =
ButtonDefaults.buttonColors(containerColor = Color(0xFF81B29A)),
                                    shape =
RoundedCornerShape(12.dp),
                                    modifier =
Modifier.weight(1f)
                                ) {
                                    Text("Info Detail", color =
Color.White, fontSize = 12.sp)
                                }

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

                                Button(
                                    onClick = {
                                        Log.d("DinoListScreen",
"Tombol Wikipedia ditekan untuk: ${dino.name}")
16                                      val intent =
Intent(Intent.ACTION_VIEW, Uri.parse(dino.wikiUrl))

context.startActivity(intent)
```

```
                                    },
                                    colors =
ButtonDefaults.buttonColors(containerColor = Color(0xFFE07A5F)),
                                    shape =
RoundedCornerShape(12.dp),
                                    modifier =
Modifier.weight(1f)
                                ) {
                                    Text("Wikipedia", color =
Color.White, fontSize = 12.sp)
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}


@Composable
fun DropdownMenuBox(
    options: List<String>,
    selectedOption: String,
    onOptionSelected: (String) -> Unit
) {
    var expanded by remember { mutableStateOf(false) }

    Box {
        Button(
            onClick = { expanded = true },
            colors = ButtonDefaults.buttonColors(containerColor =
Color(0xFF81B29A))
```

```
202             ) {
203                 Text(selectedOption, color = Color.White)
204             }
205
206             DropdownMenu(
207                 expanded = expanded,
208                 onDismissRequest = { expanded = false }
209             ) {
210                 options.forEach { label ->
211                     DropdownMenuItem(
212                         text = { Text(label) },
213                         onClick = {
214                             onOptionSelected(label)
215                             expanded = false
216                         }
217                     )
218                 }
219             }
220         }
221 }
```

- DinoViewModel.kt

Table 20. Sourcecode DinoViewModel Soal 1 Modul 4

```
1  package com.example.modul4.presentation.viewmodel
2
3  import android.util.Log
4  import androidx.lifecycle.ViewModel
5  import androidx.lifecycle.viewModelScope
6  import com.example.modul4.data.repository.DinoRepositoryImpl
7  import com.example.modul4.domain.model.Dino
8  import kotlinx.coroutines.flow.MutableStateFlow
9  import kotlinx.coroutines.flow.StateFlow
10 import kotlinx.coroutines.launch
11
12
13 class DinoViewModel : ViewModel() {
14
15     private val repository = DinoRepositoryImpl()
16
17     private val _dinoList =
   MutableStateFlow<List<Dino>>(emptyList())
18     val dinoList: StateFlow<List<Dino>> = _dinoList
19
20     init {
21         viewModelScope.launch {
22             repository.getAllDinos().collect {
23                 _dinoList.value = it
24                 Log.d("DinoViewModel", "Data dinosaurus berhasil
   dimuat: ${it.size} item.")
25             }
26         }
27     }
28
29
30
31
```

```
32      private val _selectedDino = MutableStateFlow<Dino?>(null)
33      val selectedDino: StateFlow<Dino?> = _selectedDino
34      fun selectDino(dino: Dino) {
35          _selectedDino.value = dino
36          Log.d("DinoViewModel", "Dino dipilih: ${dino.name}")
37      }
38  }
39
40
41
```

- DinoViewModelFactory.kt

Table 21. Sourcecode DinoViewModelFactory Soal 1 Modul 4

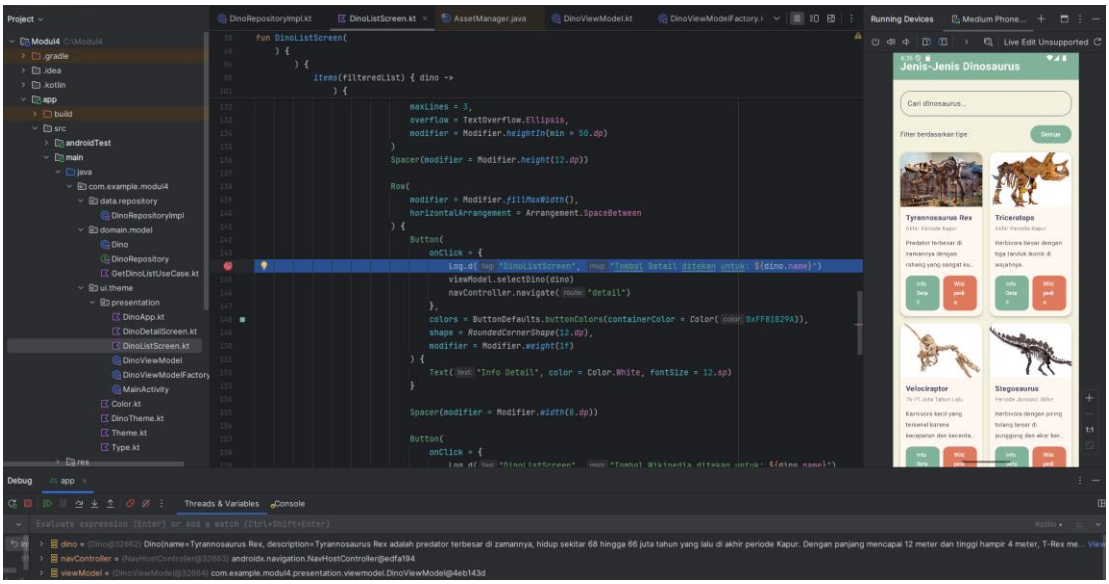
```
1   package com.example.modul4.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.ViewModelProvider
5
6   class DinoViewModelFactory : ViewModelProvider.Factory {
7       override fun <T : ViewModel> create(modelClass: Class<T>): T
8   {
9           if
10  (modelClass.isAssignableFrom(DinoViewModel::class.java)) {
11              @Suppress("UNCHECKED_CAST")
12              return DinoViewModel() as T
13          }
14          throw IllegalArgumentException("Unknown ViewModel class")
15      }
16  }
17
18
```

- MainActivity.kt

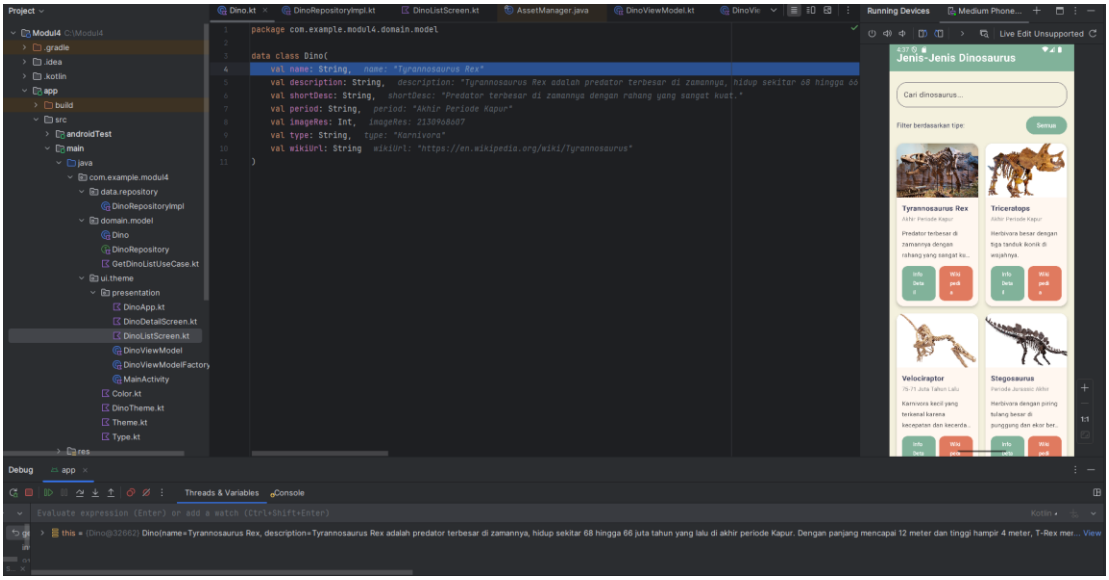
Table 22. Sourcecode MainActivity Soal 1 Modul 4

```
1  package com.example.modul4.presentation
2
3  import android.os.Bundle
4  import com.example.modul4.ui.theme.presentation.DinoApp
5  import androidx.activity.ComponentActivity
6  import androidx.activity.compose.setContent
7  import com.example.modul4.ui.theme.DinoTheme
8
9  class MainActivity : ComponentActivity() {
10     override fun onCreate(savedInstanceState: Bundle?) {
11         super.onCreate(savedInstanceState)
12         setContent {
13             DinoTheme {
14                 DinoApp()
15             }
16         }
17     }
18 }
```

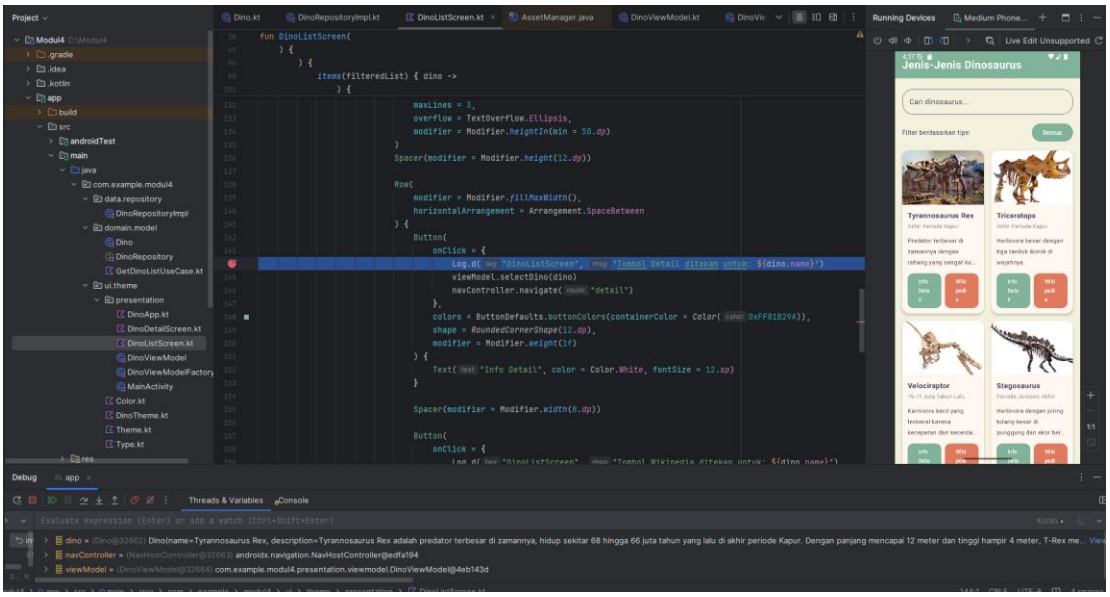
B. Output Program



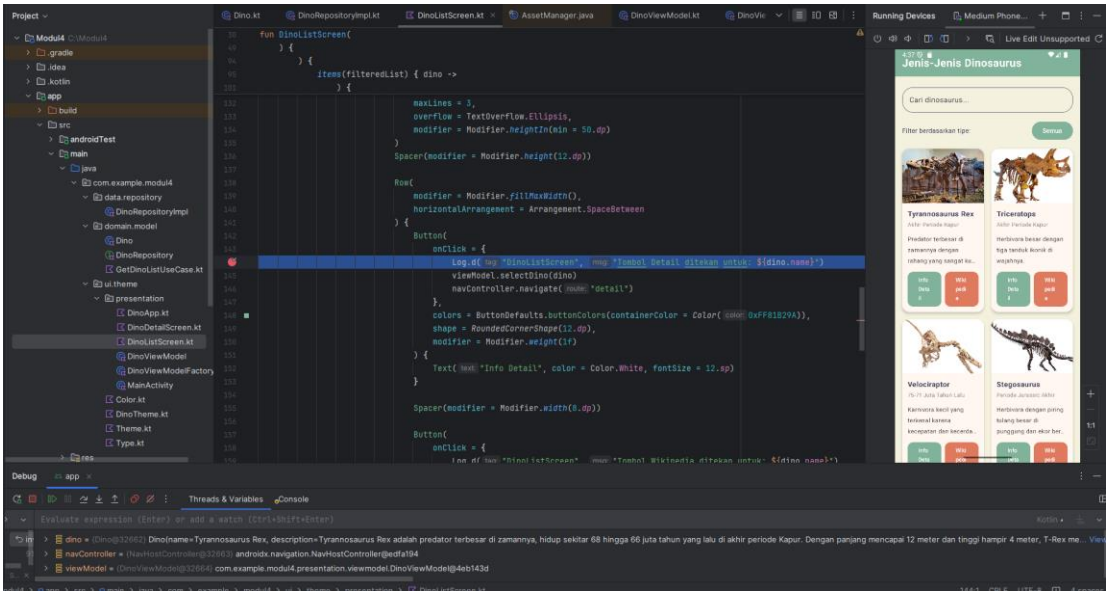
Gambar 15. Screenshot Hasil Jawaban Soal 1 Modul 4



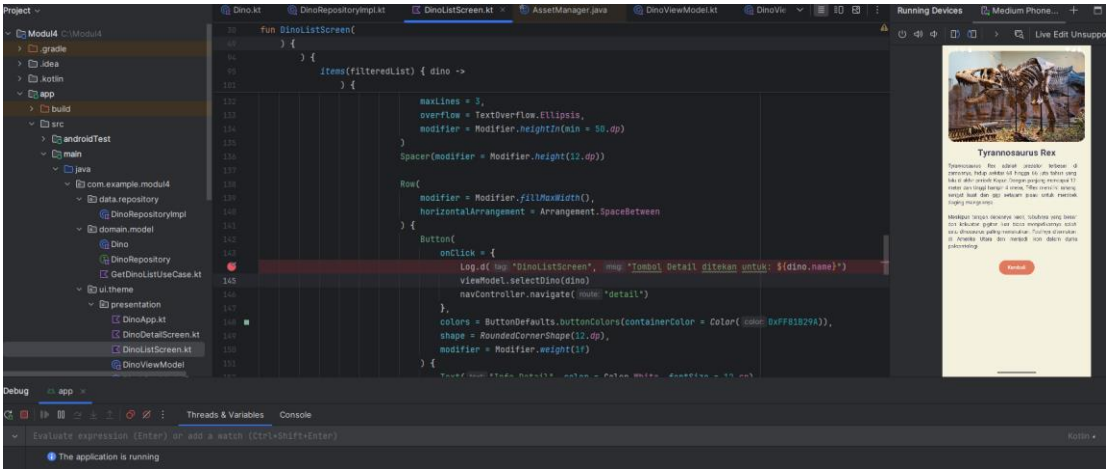
Gambar 16. Screenshot Hasil Jawaban Soal 1 Modul 4



Gambar 17. Screenshot Hasil Jawaban Soal 1 Modul 4



Gambar 18. Screenshot Hasil Jawaban Soal 1 Modul 4



Gambar 19. Screenshot Hasil Jawaban Soal 1 Modul 4

C. Pembahasan

- DinoRepositoryImpl.kt:

  Kelas DinoRepositoryImpl berisi logika untuk menyediakan data dinosaurus. Di dalamnya, terdapat sebuah properti privat bernama dinos yang merupakan sebuah List statis (hardcoded) berisi objek-objek Dino. Kelas ini juga mengimplementasikan dua fungsi *interface* yaitu getAllDinos() yang membungkus seluruh daftar dino ke dalam sebuah Flow menggunakan flowOf(), dan searchDinos(query: String) yang memfilter daftar dino berdasarkan nama yang mengandung query (tanpa mempedulikan huruf besar/kecil) sebelum membungkus hasilnya ke dalam Flow.

- Dino.kt

  Kelas ini berisikan 5 kelas yang dimana isinya ada nama(string), deskripsi(string), imageRes(integer) dan tipe(string) dan wikiUrl(string) bertujuan untuk menyimpan data

- DinoRepository.kt

  DinoRepository menetapkan dua fungsi yaitu *getAllDinos()* yang bertugas mengembalikan *Flow<List<Dino>>*, dan *searchDinos(query: String)* yang juga mengembalikan *Flow<List<Dino>>*. Penggunaan Flow dari Kotlin Coroutines memungkinkan data untuk diterima secara *asynchronous* sebagai aliran data (stream)

- SearchDinosUseCase.kt

  Bertujuan untuk mencari dinosaurus berdasarkan nama, file ini menerima query pencarian, lalu memanggil fungsi searchDinos dari repository untuk menjalankan tugasnya dan mengembalikan hasilnya.

- DinoApp.kt

  mengatur struktur utama aplikasi dan sistem navigasi. Di dalamnya, *rememberNavController()* digunakan untuk membuat NavController, yang bertugas mengelola perpindahan antar layar. Dan viewmodel disana menggunakan *viewModel(factory = DinoViewModelFactory())* untuk navigasi. Dan NavHost berguna sebagai navigasi pada semua layar

- DinoDetailScreen.kt

  Disini bertujuan untuk menampilkan informasi secara lengkap dari dinosaurus yang dipilih, dan data nya diambil melalui SelectDino yang tersimpan dalam ViewModel, lalu ada Column agar semua informasi tersusun rapi dan ada verticalScroll agar bisa scroll. Lalu

juga saya meanmbahkan tombol navigasi kembali dengan cara navController.popBackStack()

- DinoListScreen.kt

Data dinoList diambil dari ViewModel dan diubah menjadi *state* yang dapat diamati oleh Compose menggunakan collectAsState(). Lalu pada halaman ini ada OutlinedTextField yang memungkinkan pengguna mengetik searchQuery untuk mencari dinosaurus. Terdapat juga DropdownMenuBox untuk memfilter daftar berdasarkan selectedFilter ("Semua", "Herbivora", atau "Karnivora"). Lalu pada tiap card terdapat navigasi “Info Detail” dan “Wikipedia”

- DinoViewModel.kt

DinoViewModel berfungsi sebagai jembatan antara lapisan data dan lapisan UI. Di dalamnya terdapat instance dari DinoRepositoryImpl untuk mengambil data. Lalu ada dua StateFlow yaitu _dinoList yang menyimpan daftar dinosaurus dan _selectedDino yang menyimpan dinosaurus yang sedang dipilih oleh pengguna.

- DinoViewModelFactory.kt

File ini bertujuan untuk mengontrol bagaimana DinoViewModel. Metode create di dalamnya akan memeriksa apakah kelas yang diminta adalah DinoViewModel. Jika ya, ia akan membuat dan mengembalikan instance baru dari DinoViewModel; jika tidak, ia akan melemparkan IllegalArgumentException untuk mencegah kesalahan.

- MainActivity.kt

  File ini adalah titik masuk dari aplikasi android. Jadi pada file ini terdapat metode setContent yanf berguna untuk mendefinisikan tata letak UI aplikasi menggunakan fungsi-fungsi Composable. Di sini, setContent memanggil Composable DinoApp() yang dibungkus di dalam DinoTheme, yang bertanggung jawab untuk menerapkan tema visual (seperti warna dan font) ke seluruh aplikasi.

1. E. Debugger (Step Into, Step Over, Step Out)

Debugger adalah alat di Android Studio yang memungkinkan Anda menjalankan aplikasi baris per baris untuk memeriksa nilai variabel dan alur eksekusi kode. Ini sangat penting untuk menemukan dan memperbaiki bug.

Cara Menggunakanya:

- Menggunakan Breakpoint: Klik pada area di sebelah kiri nomor baris kode, Ini adalah *breakpoint*, titik di mana eksekusi aplikasi akan berhenti sejenak saat mode debug dijalankan.

- Mulai Debug: Jalankan aplikasi dengan menekan ikon kumbang atau melalui menu Run > Debug 'app'.

- Saat eksekusi mencapai *breakpoint*, jendela Debug akan muncul di bagian bawah, menampilkan variabel saat ini dan *call stack*.

- **Step Over (F8)**: Mengeksekusi baris kode saat ini dan pindah ke baris berikutnya dalam fungsi yang sama. Dan Step Over akan mengeksekusi seluruh fungsi tersebut tanpa masuk ke dalamnya.

- **Step Into (F7):** Step Into akan masuk ke dalam fungsi tersebut dan berhenti di baris pertama di dalamnya. Ini bertujuan untuk melihat apa yang terjadi di dalam fungsi yang Anda panggil.

- **Step Out (Shift + F8):** Mengeksekusi sisa baris kode di dalam fungsi saat ini dan keluar.

2. **Application Class** adalah sebuah kelas dasar dalam aplikasi Android yang diinisialisasi sebelum komponen lain (seperti `Activity` atau `Service`) dibuat saat proses aplikasi dimulai. Setiap aplikasi hanya memiliki satu instance `Application`.

   Fungsi Utama:

- **Inisialisasi Global**: Tempat terbaik untuk menginisialisasi library atau state yang bersifat global dan perlu ada selama siklus hidup aplikasi. Contoh: inisialisasi library logging (seperti Timber), dependency injection (seperti Hilt atau Koin), atau analytics.
- **Mengelola State Global**: Meskipun tidak disarankan untuk menyimpan data yang berhubungan dengan UI, kelas ini bisa digunakan untuk mengelola state global yang tidak terikat pada `Activity` tertentu.
- **Lifecycle Callbacks**: Anda dapat me-listen *lifecycle events* dari komponen aplikasi lain.

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 untuk status dan error handling pada API dan Flow untuk data stream.
   b. Gunakan KotlinX Serialization sebagai library JSON.
   c. Gunakan library seperti Coil atau Glide untuk image loading.
   d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started
   e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
   f. Gunakan caching strategy pada Room..
   g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.
   Aplikasi harus mempertahankan fitur-fitur yang dibuat pada modul sebelumnya

A. Source Code

- DinoApiService

Table 23. Sourcecode DinoApiService Soal 1 Modul 5

```
1  package com.example.modul5.data
2
   import com.example.modul5.data.model.DinoDto
3  import retrofit2.http.GET
4
   interface DinoApiService {
5      @GET("api/dinosaurs")
       suspend fun getDinos(): List<DinoDto>
6  }
7
```

- DinoDao.kt

Table 24 Sourcecode DinoDao soal 1 Modul 5

```
1  package com.example.modul5.data
2
   import androidx.room.Dao
3  import androidx.room.Insert
   import androidx.room.OnConflictStrategy
4  import androidx.room.Query
5
   @Dao
6  interface DinoDao {
7      @Query("SELECT * FROM dinos")
       suspend fun getAll(): List<DinoEntity>
8
       @Insert(onConflict = OnConflictStrategy.REPLACE)
```

```
9       suspend fun insertAll(dinos: List<DinoEntity>)
10
        @Query("DELETE FROM dinos")
11      suspend fun clear()
    }
12
```

- DinoDatabse.kt

Table 25 Sourcecode DinoDatabase Soal 1 Modul 5

```
1   package com.example.modul5.data
2
    import android.content.Context
3   import androidx.room.Database
    import androidx.room.Room
4   import androidx.room.RoomDatabase
5
    @Database(entities = [DinoEntity::class], version = 1)
6   abstract class DinoDatabase : RoomDatabase() {
7       abstract fun dinoDao(): DinoDao
8
        companion object {
9           @Volatile private var INSTANCE: DinoDatabase? = null
10
            fun getInstance(context: Context): DinoDatabase =
11              INSTANCE ?: synchronized(this) {
                    INSTANCE ?: Room.databaseBuilder(
12                      context.applicationContext,
13                      DinoDatabase::class.java,
                        "dino.db"
14                  ).build().also { INSTANCE = it }
                }
```

```
15      }
    }
16
17
```

- DinoDto.kt

Table 26 Sourcecode DinoDto Soal 1 Modul 5

```
1   package com.example.modul5.data.model

2
    import kotlinx.serialization.SerialName
3   import kotlinx.serialization.Serializable
    import com.example.modul5.domain.Dino
4
5   @Serializable
    data class DinoDto(
6       val name: String,
7       val description: String,
        val diet: String,
8       val period: String,
9       val image: String
    ) {
10      fun toDomain() = Dino(name, description, image, diet, period)
11  }

12
```

- DinoEntity.kt

Table 27 Sourcecode DinoEntity Soal 1 Modul 5

```
1  package com.example.modul5.data
2
3  import androidx.room.Entity
4  import androidx.room.PrimaryKey
5  import com.example.modul5.domain.Dino
6
7  @Entity(tableName = "dinos")
8  data class DinoEntity(
9      @PrimaryKey val name: String,
10     val description: String,
11     val imageUrl: String,
12     val diet: String,
13     val period: String
14 )
15
16 fun DinoEntity.toDomain() = Dino(
17     name, description, imageUrl, diet, period
18 )
19
20 fun Dino.toEntity() = DinoEntity(
21     name, description, imageUrl, diet, period
22 )
```

- DinorepositoryImpl.kt

Table 28 Sourcecode DinoRepositoryImpl Soal 1 Modul 5

```
1  package com.example.modul5.data.repository
2
3  import com.example.modul5.data.DinoApiService
4  import com.example.modul5.data.DinoDao
5  import com.example.modul5.data.Resource
6  import com.example.modul5.data.toDomain
7  import com.example.modul5.data.toEntity
8  import com.example.modul5.domain.Dino
9  import com.example.modul5.domain.repository.DinoRepository
10 import kotlinx.coroutines.Dispatchers
11 import kotlinx.coroutines.flow.Flow
12 import kotlinx.coroutines.flow.MutableStateFlow
13 import kotlinx.coroutines.flow.flow
14 import kotlinx.coroutines.flow.flowOn
15 import retrofit2.HttpException
16 import java.io.IOException
17
18 class DinoRepositoryImpl(
19     private val apiService: DinoApiService,
20     private val dao: DinoDao
21 ) : DinoRepository {
22
23     private val selectedDino = MutableStateFlow<Dino?>(null)
24
25     override fun getAllDinos(): Flow<Resource<List<Dino>>> = flow
26 {
27         emit(Resource.Loading())
28
29         val cachedDinos = dao.getAll().map { it.toDomain() }
30         emit(Resource.Success(cachedDinos))
31
```

```
28          try {
29              val remoteDinos = apiService.getDinos()
30              dao.clear()
31              dao.insertAll(remoteDinos.map {
32  it.toDomain().toEntity() })
33          } catch (e: HttpException) {
34
35              emit(Resource.Error(
36                  "Terjadi kesalahan koneksi. Menampilkan data
37  offline.",
38                  cachedDinos
39              ))
40          } catch (e: IOException) {
41
42              emit(Resource.Error(
43                  "Tidak ada internet. Menampilkan data offline.",
44                  cachedDinos
45              ))
46          }
47
48          val newDinos = dao.getAll().map { it.toDomain() }
49          emit(Resource.Success(newDinos))
50
51      }.flowOn(Dispatchers.IO)
52
53      override fun selectDino(dino: Dino) {
54          selectedDino.value = dino
55      }
56
57      override fun getSelectedDino(): Dino? = selectedDino.value
58  }
```

- Resource.kt

Table 29 Sourcecode Resource Soal 1 Modul 5

```
1  package com.example.modul5.data
2
   sealed class Resource<T>(val data: T? = null, val message: String?
3  = null) {
       class Success<T>(data: T) : Resource<T>(data)
4
       class Error<T>(message: String, data: T? = null) :
5  Resource<T>(data, message)
       class Loading<T>(data: T? = null) : Resource<T>(data)
6  }
7
```

- DinoViewmodel.kt

Table 30 Sourcecode DinoViewModel Soal 1 Modul 5

```
1  package com.example.modul5.data
2
3  import com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
4  import kotlinx.serialization.json.Json
5  import okhttp3.MediaType.Companion.toMediaType
6  import retrofit2.Retrofit
7  import com.example.modul5.data.DinoApiService
8  import com.example.modul5.data.model.DinoDto
9
10 object RetrofitInstance {
11     private val json = Json { ignoreUnknownKeys = true }
12     private val contentType = "application/json".toMediaType()
13
14     val api: DinoApiService by lazy {
15         Retrofit.Builder()
16             .baseUrl("https://dinoapi.brunosouzadev.com/")
17             .addConverterFactory(json.asConverterFactory(contentType))
18             .build()
             .create(DinoApiService::class.java)
     }
}
```

- ThemePreference.kt

Table 31. Sourcecode ThemePreference Soal 1 Modul 5

```
1  package com.example.modul5.data
2
   import android.content.Context
3  import androidx.datastore.preferences.core.booleanPreferencesKey
   import androidx.datastore.preferences.core.edit
4  import androidx.datastore.preferences.preferencesDataStore
5  import kotlinx.coroutines.flow.Flow
   import kotlinx.coroutines.flow.map
6
7  val Context.dataStore by preferencesDataStore("settings")
8
   class ThemePreferences(private val context: Context) {
9      companion object {
           private val DARK_MODE_KEY =
10 booleanPreferencesKey("dark_mode")
11     }
12
       val darkModeFlow: Flow<Boolean> = context.dataStore.data
13         .map { preferences -> preferences[DARK_MODE_KEY] ?: false
   }
14
15     suspend fun saveDarkModeSetting(isDarkMode: Boolean) {
           context.dataStore.edit { preferences ->
16             preferences[DARK_MODE_KEY] = isDarkMode
           }
17     }
18 }
19
20
```

-

- Dino.kt

Table 32. Sourcecode Dino Soal 1 Modul 5

```
1  package com.example.modul5.domain
2
   data class Dino(
3      val name: String,
4      val description: String,
       val imageUrl: String,
5      val diet: String,
6      val period: String
   )
7
```

- Dinorepository.kt

Table 33. Sourcecode DinoRepository Soal 1 Modul 5

```
1  package com.example.modul5.domain.repository
2
   import com.example.modul5.data.Resource
3  import com.example.modul5.domain.Dino
4  import kotlinx.coroutines.flow.Flow
5
   interface DinoRepository {
6      fun getAllDinos(): Flow<Resource<List<Dino>>>
7
       fun selectDino(dino: Dino)
8      fun getSelectedDino(): Dino?
   }
9
```

- DinoDetailScreen.kt

Table 34. Sourcecode DinoDetailScreen Soal 1 Modul 5

```
1   package com.example.modul5.presentation.ui
2
    import androidx.compose.foundation.layout.*
3   import androidx.compose.foundation.rememberScrollState
4   import androidx.compose.foundation.verticalScroll
    import androidx.compose.material3.*
5   import androidx.compose.runtime.Composable
6   import androidx.compose.runtime.collectAsState
    import androidx.compose.ui.Alignment
7   import androidx.compose.ui.Modifier
8   import androidx.compose.ui.draw.clip
9   import androidx.compose.ui.layout.ContentScale
    import androidx.compose.ui.text.font.FontWeight
10  import androidx.compose.ui.text.style.TextAlign
11  import androidx.compose.ui.unit.dp
    import androidx.compose.ui.unit.sp
12  import androidx.navigation.NavController
13  import coil.compose.AsyncImage
    import androidx.compose.foundation.shape.RoundedCornerShape
14  import com.example.modul5.presentation.viewmodel.DinoViewModel
15
16  @Composable
    fun DinoDetailScreen(viewModel: DinoViewModel, navController:
17  NavController) {
188     val selectedDino =
    viewModel.selectedDino.collectAsState().value
19
20      if (selectedDino == null) {
            Text("Data tidak tersedia")
21          return
22      }
```

```
23
    Column(
24      modifier = Modifier
            .padding(16.dp)
25          .verticalScroll(rememberScrollState())
            .fillMaxSize()
26  ) {
        AsyncImage(
27          model = selectedDino.imageUrl,
            contentDescription = null,
28          contentScale = ContentScale.Crop,
            modifier = Modifier
29              .fillMaxWidth()
                .height(250.dp)
30              .clip(RoundedCornerShape(20.dp))
        )
31      Spacer(modifier = Modifier.height(16.dp))
        Text(
32          selectedDino.name,
            fontSize = 22.sp,
33          fontWeight = FontWeight.Bold,
            modifier = Modifier.fillMaxWidth(),
34          textAlign = TextAlign.Center
        )
35      Spacer(modifier = Modifier.height(12.dp))
        Text(
36          selectedDino.description,
            fontSize = 14.sp,
37          textAlign = TextAlign.Justify
        )
38      Spacer(modifier = Modifier.height(16.dp))
        Button(
39          onClick = { navController.navigate("dino_list") },
            modifier =
40
```

```
47  Modifier.align(Alignment.CenterHorizontally)
48          ) {
                Text("Kembali")
49          }
50      }
    }
51
52
```

- DinoListScreen.kt

Table 35. SourceCode DinoListScreen Soal 1 Modul 5

```
1   package com.example.modul5.presentation.ui
2
    import android.widget.Toast
3   import androidx.compose.foundation.background
4   import androidx.compose.foundation.layout.*
    import androidx.compose.foundation.lazy.grid.GridCells
5   import androidx.compose.foundation.lazy.grid.LazyVerticalGrid
6   import androidx.compose.foundation.lazy.grid.items
    import androidx.compose.foundation.shape.RoundedCornerShape
7   import androidx.compose.material.icons.Icons
8   import androidx.compose.material.icons.filled.DarkMode
    import androidx.compose.material.icons.filled.LightMode
9   import androidx.compose.material3.*
10  import androidx.compose.runtime.*
11  import androidx.compose.ui.Alignment
    import androidx.compose.ui.Modifier
12  import androidx.compose.ui.draw.clip
13  import androidx.compose.ui.layout.ContentScale
    import androidx.compose.ui.platform.LocalContext
14  import androidx.compose.ui.text.style.TextAlign
15  import androidx.compose.ui.unit.dp
```

```
import androidx.compose.ui.unit.sp
import androidx.navigation.NavController
import coil.compose.AsyncImage
import com.example.modul5.data.Resource
import com.example.modul5.presentation.viewmodel.DinoViewModel
import com.example.modul5.presentation.viewmodel.SettingsViewModel
import com.google.accompanist.systemuicontroller.rememberSystemUiController

@Composable
fun DinoListScreen(
    navController: NavController,
    viewModel: DinoViewModel,
    settingsViewModel: SettingsViewModel,
    isDarkMode: Boolean
) {
    val context = LocalContext.current
    var searchQuery by remember { mutableStateOf("") }
    var selectedDiet by remember { mutableStateOf("All") }
    var expanded by remember { mutableStateOf(false) }

    val dinoListState by viewModel.dinoListState.collectAsState()
    val dietOptions = listOf("All", "Herbivora", "Karnivora", "Omnivora")
    val systemUiController = rememberSystemUiController()
    val statusBarColor = MaterialTheme.colorScheme.primary
    val colorScheme = MaterialTheme.colorScheme

    SideEffect {
        systemUiController.setStatusBarColor(color = statusBarColor, darkIcons = !isDarkMode)
    }
```

```
46
47      LaunchedEffect(key1 = dinoListState) {
48          if (dinoListState is Resource.Error &&
49  dinoListState.data?.isNotEmpty() == true) {
50              Toast.makeText(context, dinoListState.message,
51  Toast.LENGTH_LONG).show()
52          }
53      }
54
55      val filteredList = dinoListState.data?.filter { dino ->
56          val matchesSearch = dino.name.contains(searchQuery,
57  ignoreCase = true)
58          val apiDiet = when (selectedDiet) {
59              "Herbivora" -> "herbivore"
60              "Karnivora" -> "carnivore"
61              "Omnivora" -> "omnivore"
62              else -> "All"
63          }
64          val matchesDiet = apiDiet == "All" ||
65  dino.diet.equals(apiDiet, ignoreCase = true)
66          matchesSearch && matchesDiet
67      } ?: emptyList()
68
69      Column(modifier =
70  Modifier.background(colorScheme.background).fillMaxSize()) {
71
72          Row(
73              modifier = Modifier
74                  .fillMaxWidth()
75                  .background(colorScheme.primary)
76                  .padding(16.dp),
77              verticalAlignment = Alignment.CenterVertically,
78              horizontalArrangement = Arrangement.SpaceBetween
79          ) {
```

```
80              Text(
81                  text = "Jenis-Jenis Dinosaurus",
82                  fontSize = 24.sp,
83                  color = colorScheme.onPrimary,
84              )
85              IconButton(onClick = {
86  settingsViewModel.toggleDarkMode() }) {
87                  Icon(
88                      imageVector = if (isDarkMode)
89  Icons.Default.LightMode else Icons.Default.DarkMode,
90                      contentDescription = "Toggle Dark Mode",
91                      tint = colorScheme.onPrimary
92                  )
93              }
94          }
95
96
97          OutlinedTextField(
98              value = searchQuery,
99              onValueChange = { searchQuery = it },
100             label = { Text("Cari dinosaurus...") },
101             shape = RoundedCornerShape(24.dp),
102             modifier = Modifier
103                 .fillMaxWidth()
104                 .padding(16.dp),
105             colors = OutlinedTextFieldDefaults.colors(
106                 focusedBorderColor = colorScheme.primary,
107                 unfocusedBorderColor = colorScheme.outline,
108                 cursorColor = colorScheme.primary
109             )
110         )
111
112
113         Box(
```

```
            modifier = Modifier
                .fillMaxWidth()
                .padding(horizontal = 16.dp)
        ) {
            OutlinedButton(onClick = { expanded = true }) {
                Text("Filter: $selectedDiet")
            }
            DropdownMenu(expanded = expanded, onDismissRequest =
{ expanded = false }) {
                dietOptions.forEach { option ->
                    DropdownMenuItem(
                        text = { Text(option) },
                        onClick = {
                            selectedDiet = option
                            expanded = false
                        }
                    )
                }
            }
        }


        Box(
            modifier = Modifier
                .fillMaxSize()
                .padding(top = 8.dp),
            contentAlignment = Alignment.Center
        ) {
            LazyVerticalGrid(
                columns = GridCells.Fixed(2),
                contentPadding = PaddingValues(horizontal =
16.dp, vertical = 8.dp),
                verticalArrangement =
Arrangement.spacedBy(12.dp),
```

```
                horizontalArrangement =
Arrangement.spacedBy(12.dp)
            ) {
                items(filteredList) { dino ->
                    Card(
                        modifier = Modifier.fillMaxWidth(),
                        colors =
CardDefaults.cardColors(containerColor = colorScheme.surface),
                        elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp),
                        shape = RoundedCornerShape(16.dp)
                    ) {
                        Column {
                            AsyncImage(
                                model = dino.imageUrl,
                                contentDescription = dino.name,
                                contentScale = ContentScale.Crop,
                                modifier = Modifier
                                    .fillMaxWidth()
                                    .height(140.dp)

.clip(RoundedCornerShape(topStart = 16.dp, topEnd = 16.dp))
                            )
                            Column(modifier =
Modifier.padding(12.dp)) {
                                Text(
                                    dino.name,
                                    fontSize = 16.sp,
                                    color = colorScheme.onSurface
                                )
                                Spacer(modifier =
Modifier.height(8.dp))
                                Button(
                                    onClick = {
```

```
182
183 viewModel.selectDino(dino)
184
185 navController.navigate("detail")
186                                     },
187                                     modifier =
188 Modifier.fillMaxWidth(),
189                                     colors =
190 ButtonDefaults.buttonColors(containerColor = colorScheme.primary)
191                                 ) {
192                                     Text("Info Detail", color =
193 colorScheme.onPrimary, fontSize = 12.sp)
194                                 }
195                             }
196                         }
197                     }
198                 }
199             }
200
201             if (dinoListState is Resource.Loading &&
202 filteredList.isEmpty()) {
203                 CircularProgressIndicator()
204             }
205
206             if (dinoListState is Resource.Error &&
207 filteredList.isEmpty()) {
208                 Text(
209                     text = dinoListState.message ?: "Gagal memuat
210 data",
211                     color = colorScheme.error,
212                     textAlign = TextAlign.Center,
213                     modifier = Modifier.padding(16.dp)
214                 )
215             }
```

```
216             }
217         }
    }
```

- MainActivity

Table 36. SourceCode MainActivity Soal 1 Modul 5

```
1   package com.example.modul5.presentation.ui
2
    import android.os.Bundle
3   import androidx.activity.ComponentActivity
4   import androidx.activity.compose.setContent
    import androidx.activity.enableEdgeToEdge
5   import androidx.compose.foundation.layout.padding
6   import androidx.compose.material3.Scaffold
    import androidx.compose.runtime.collectAsState
7   import androidx.compose.runtime.getValue
8   import androidx.compose.ui.Modifier
    import androidx.lifecycle.viewmodel.compose.viewModel
9   import androidx.navigation.compose.NavHost
10  import androidx.navigation.compose.composable
11  import androidx.navigation.compose.rememberNavController
    import com.example.modul5.presentation.theme.Modul5Theme
12  import com.example.modul5.presentation.viewmodel.DinoViewModel
13  import
    com.example.modul5.presentation.viewmodel.DinoViewModelFactory
14  import com.example.modul5.presentation.viewmodel.SettingsViewModel
15  import androidx.compose.ui.platform.LocalContext
16
    class MainActivity : ComponentActivity() {
17      override fun onCreate(savedInstanceState: Bundle?) {
188         super.onCreate(savedInstanceState)
            enableEdgeToEdge()
```

```
19          setContent {
20              val context = LocalContext.current
21              val navController = rememberNavController()
22              val viewModel: DinoViewModel = viewModel(factory =
DinoViewModelFactory(context))
23              val settingsViewModel: SettingsViewModel = viewModel()
24              val isDarkMode by
settingsViewModel.darkMode.collectAsState(initial = false)
25
26              Modul5Theme(darkTheme = isDarkMode) {
27                  Scaffold { padding ->
28                      NavHost(
29                          navController = navController,
30                          startDestination = "dino_list",
31                          modifier = Modifier.padding(padding)
32                      ) {
33                          composable("dino_list") {
34                              DinoListScreen(navController,
viewModel, settingsViewModel, isDarkMode)
35                          }
36                          composable("detail") {
37
38                              DinoDetailScreen(viewModel,
navController)
39                          }
40                      }
41                  }
              }
          }
      }
```

- DinoViewModel.kt

Table 37. SourceCode DinoViewModel Soal 1 Modul 5

```
1  package com.example.modul5.presentation.viewmodel
2
3  import android.content.Context
4  import androidx.lifecycle.ViewModel
5  import androidx.lifecycle.ViewModelProvider
6  import com.example.modul5.data.DinoDatabase
7  import com.example.modul5.data.RetrofitInstance
8  import com.example.modul5.data.repository.DinoRepositoryImpl
9  import com.example.modul5.domain.repository.DinoRepository
10
11
12 class DinoViewModelFactory(private val context: Context) :
13 ViewModelProvider.Factory {
14     override fun <T : ViewModel> create(modelClass: Class<T>): T
15 {
16         if
17 (modelClass.isAssignableFrom(DinoViewModel::class.java)) {
18
19             val repository: DinoRepository = DinoRepositoryImpl(
20                 apiService = RetrofitInstance.api,
21                 dao =
22 DinoDatabase.getInstance(context).dinoDao()
23             )
24             @Suppress("UNCHECKED_CAST")
25             return DinoViewModel(repository) as T
26         }
27         throw IllegalArgumentException("Unknown ViewModel
28 class")
29     }
30 }
31
```

- DinoViewModelFactory

Table 38. SourceCode DinoViewModelFactory Soal 1 Modul 5

```
1  package com.example.modul5.presentation.viewmodel
2
3  import androidx.lifecycle.ViewModel
4  import androidx.lifecycle.viewModelScope
5  import com.example.modul5.data.Resource
6  import com.example.modul5.domain.Dino
7  import com.example.modul5.domain.repository.DinoRepository
8  import kotlinx.coroutines.flow.MutableStateFlow
9  import kotlinx.coroutines.flow.StateFlow
10 import kotlinx.coroutines.flow.asStateFlow
11 import kotlinx.coroutines.flow.launchIn
12 import kotlinx.coroutines.flow.onEach
13
14 class DinoViewModel(private val repository: DinoRepository) :
15 ViewModel() {
16
17
18     private val _dinoListState =
19 MutableStateFlow<Resource<List<Dino>>>(Resource.Loading())
20     val dinoListState: StateFlow<Resource<List<Dino>>> =
21 _dinoListState.asStateFlow()
22
23
24     private val _selectedDino = MutableStateFlow<Dino?>(null)
25     val selectedDino: StateFlow<Dino?> =
26 _selectedDino.asStateFlow()
27
28     init {
29
30         getDinos()
31     }
```

```
32
33      private fun getDinos() {
34          repository.getAllDinos().onEach { result ->
35              _dinoListState.value = result
36          }.launchIn(viewModelScope)
37      }
38
39      fun selectDino(dino: Dino) {
40
41          _selectedDino.value = dino
42      }
43  }
44
45
46
```

- SettingViewModel.kt

Table 39. SourceCode SettingViewModel Soal 1 Modul 5

```
1   package com.example.modul5.presentation.viewmodel
2
3   import androidx.lifecycle.ViewModel
4   import androidx.lifecycle.viewModelScope
5   import com.example.modul5.data.Resource
6   import com.example.modul5.domain.Dino
7   import com.example.modul5.domain.repository.DinoRepository
8   import kotlinx.coroutines.flow.MutableStateFlow
9   import kotlinx.coroutines.flow.StateFlow
10  import kotlinx.coroutines.flow.asStateFlow
11  import kotlinx.coroutines.flow.launchIn
12  import kotlinx.coroutines.flow.onEach
13
14  class DinoViewModel(private val repository: DinoRepository) :
```

```
ViewModel() {


    private val _dinoListState =
MutableStateFlow<Resource<List<Dino>>>(Resource.Loading())
    val dinoListState: StateFlow<Resource<List<Dino>>> =
_dinoListState.asStateFlow()


    private val _selectedDino = MutableStateFlow<Dino?>(null)
    val selectedDino: StateFlow<Dino?> =
_selectedDino.asStateFlow()

    init {

        getDinos()
    }

    private fun getDinos() {
        repository.getAllDinos().onEach { result ->
            _dinoListState.value = result
        }.launchIn(viewModelScope)
    }

    fun selectDino(dino: Dino) {

        _selectedDino.value = dino
    }
}
```

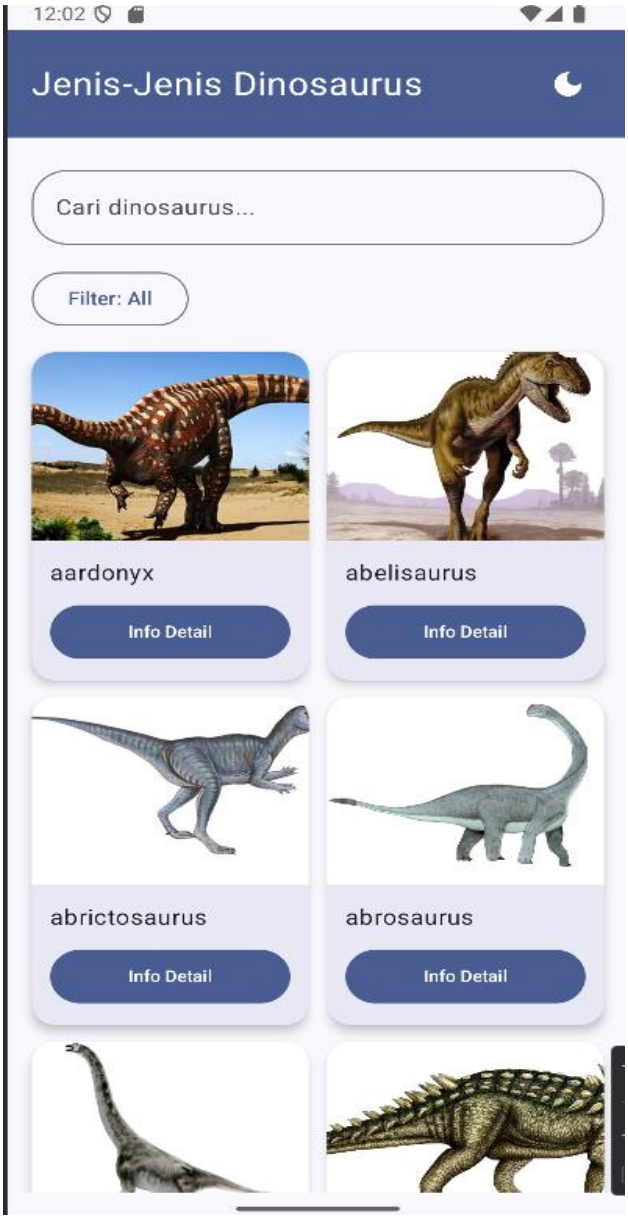
- Theme

Table 40. SourceCode Theme Soal 1 Modul 5

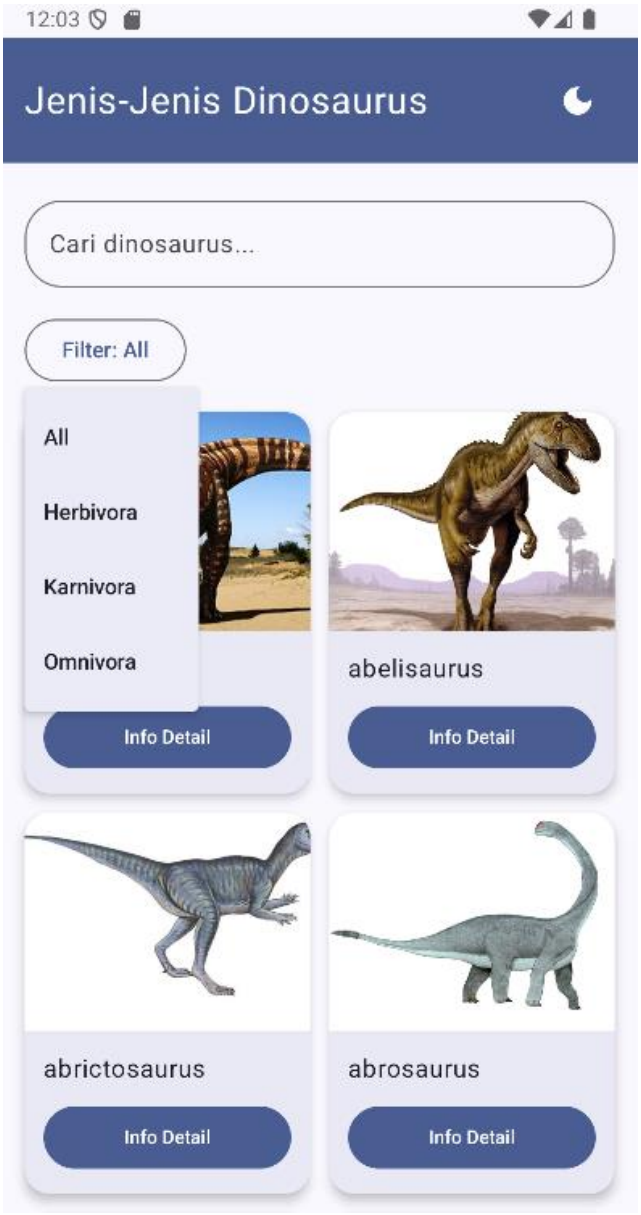
```
1  package com.example.modul5.presentation.theme
2
3  import android.app.Activity
4  import android.os.Build
5  import androidx.compose.foundation.isSystemInDarkTheme
6  import androidx.compose.material3.MaterialTheme
7  import androidx.compose.material3.darkColorScheme
8  import androidx.compose.material3.dynamicDarkColorScheme
9  import androidx.compose.material3.dynamicLightColorScheme
10 import androidx.compose.material3.lightColorScheme
11 import androidx.compose.runtime.Composable
12 import androidx.compose.ui.platform.LocalContext
13
14 private val DarkColorScheme = darkColorScheme(
15     primary = Purple80,
16     secondary = PurpleGrey80,
17     tertiary = Pink80
18 )
19
20 private val LightColorScheme = lightColorScheme(
21     primary = Purple40,
22     secondary = PurpleGrey40,
23     tertiary = Pink40
24
25
26 )
27
28 @Composable
29 fun Modul5Theme(
30     darkTheme: Boolean = isSystemInDarkTheme(),
31
```

```
    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
        else -> LightColorScheme
    }

    MaterialTheme(
        colorScheme = colorScheme,
        typography = Typography,
        content = content
    )
}
```

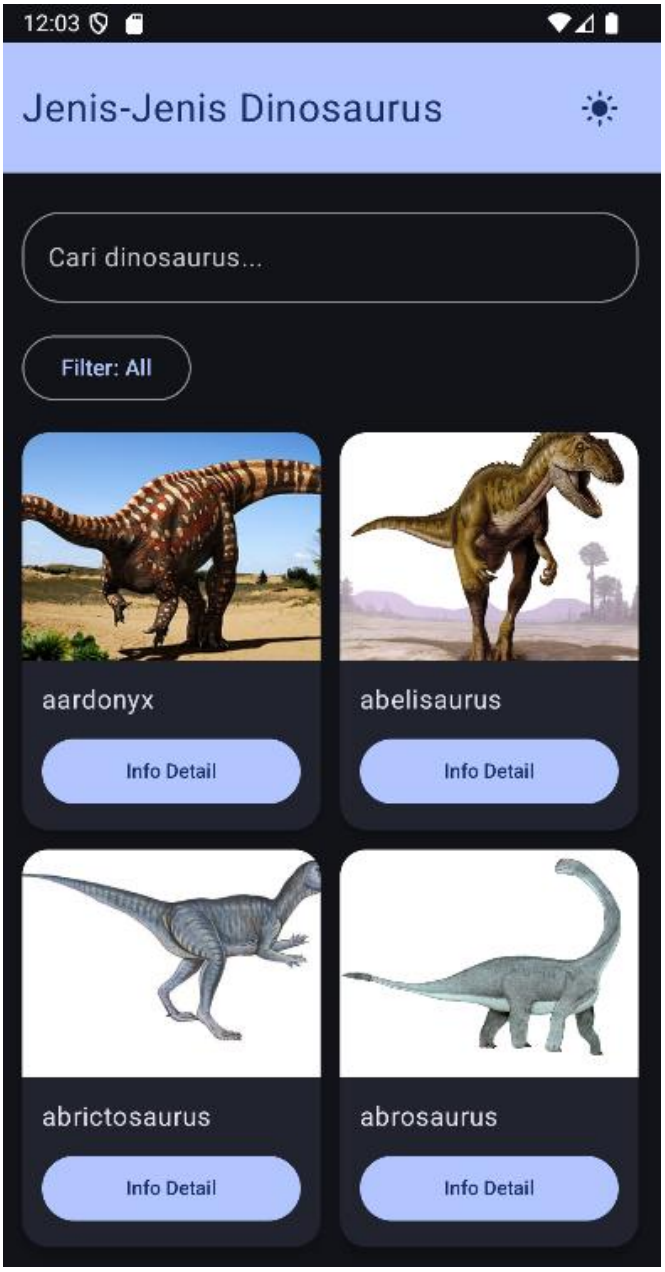
B. Output Program



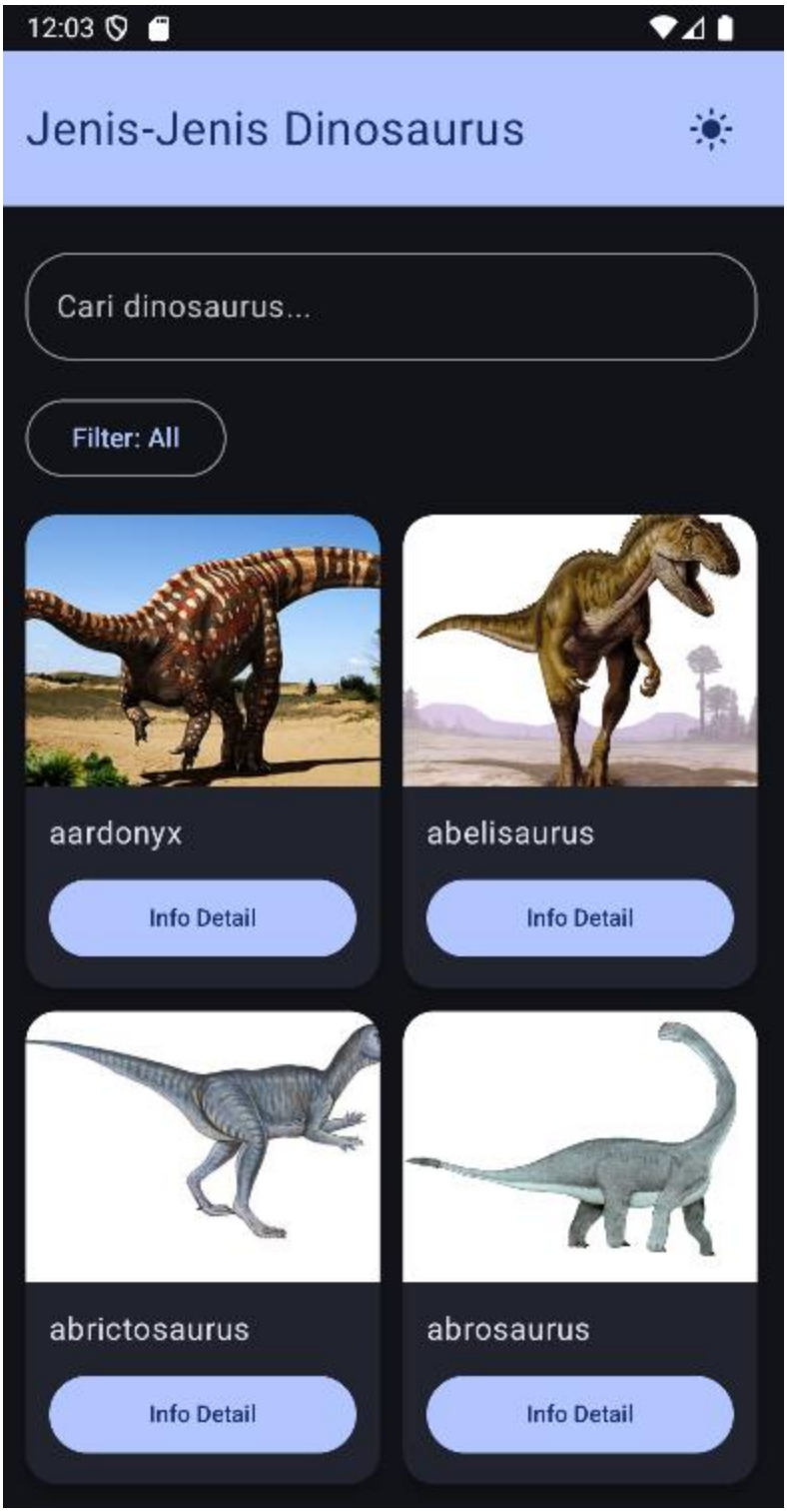
Gambar 20. Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 21. Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 22. Screenshot Hasil Jawaban Soal 1 Modul 5



Gambar 23. Screenshot Hasil Jawaban Soal 1 Modul 5

Gambar 24. Screenshot Hasil Jawaban Soal 1 Modul 5

C. Pembahasan

yang dapat di-scroll, dan terdapat tombol "Kembali" yang memanggil NavController untuk mengembalikan pengguna ke layar daftar.

- DinoViewModel

DinoViewModel.kt berfungsi sebagai pengelola *state* dan jembatan antara UI dan lapisan data. Di dalam blok init, memanggil fungsi getDinos() untuk memulai proses pengambilan data. Fungsi ini mengoleksi Flow dari repository menggunakan .launchIn(viewModelScope), dan untuk setiap Resource yang dipancarkan oleh Flow tersebut memperbarui nilai _dinoListState internalnya. *State* ini kemudian diekspos ke UI sebagai StateFlow yang *read-only*, memastikan aliran data yang searah dan terprediksi.

- DinoViewModelFactory.kt

Pada file ini bertujuan untuk membuat instance dari DinoViewModel. Kelas ini diperlukan karena DinoViewModel memiliki dependensi, yaitu DinoRepository, yang perlu disuntikkan saat dibuat. Di dalam metode create, secara manual membangun DinoRepositoryImpl dengan memberikannya semua dependensi yang dibutuhkan, seperti RetrofitInstance.api untuk jaringan dan DinoDao dari database. Repository yang sudah lengkap ini kemudian diteruskan ke dalam konstruktor DinoViewModel.

- SettingViewModel.kt

File ini bertujuan untuk mengelola logika dari pengaturan aplikasi. Lalu file ini mampu membuat instance AndroidViewModel, kemudian akan memunculkan darkModeFlow dari ThemePreferences. Dan juga didalamnya ada toggleDarkMode(), yang di dalamnya menjalankan coroutine untuk mendapatkan nilai dark mode alu memanggil fungsi saveDarkModeSetting untuk menyimpannya secara permanen.

- Theme.kt

Di dalamnya terdapat definisi konstan untuk skema warna, yaitu DarkColorScheme dan LightColorScheme. Komponen utama, Modul5Theme, menerima parameter boolean darkTheme

Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat. https://github.com/Yoruuu00/PemprogramanWebII