LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Siti Ratna Dwinta Sari **NIM. 2310817120002**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Siti Ratna Dwinta sari
NIM : 2310817120002

Menyetujui,	Mengetahui,
Asisten Praktikum	Dosen Penanggung Jawab Praktikum
Zulfa Auliya Akbar	Muti`a Maulida S.Kom M.T.I
NIM. 2210817210026	NIP. 19881027 2019032013

DAFTAR ISI

DAFTAR GAMBAR

MODUL 1: ANDROID BASIC WITH KOTLIN

MODUL 2: ANDROID LAYOUT

MODUL 3: BUILD SCROLLABLE LIST

MODUL 4: VIEW MODEL AND DEBUGGING

MODUL 5: CONNECT TO THE INTERNET

DAFTAR TABEL

MODUL 1: ANDROID BASIC WITH KOTLIN

MODUL 2: ANDROID LAYOUT

MODUL 3: BUILD SCROLLABLE LIST

MODUL 4: VIEW MODEL AND DEBUGGING

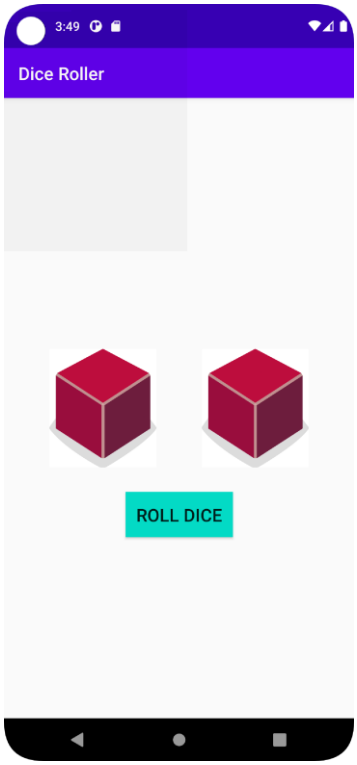
MODUL 5: CONNECT TO THE INTERNET

MODUL 1 : ANDROID BASICS WITH KOTLIN

SOAL 1

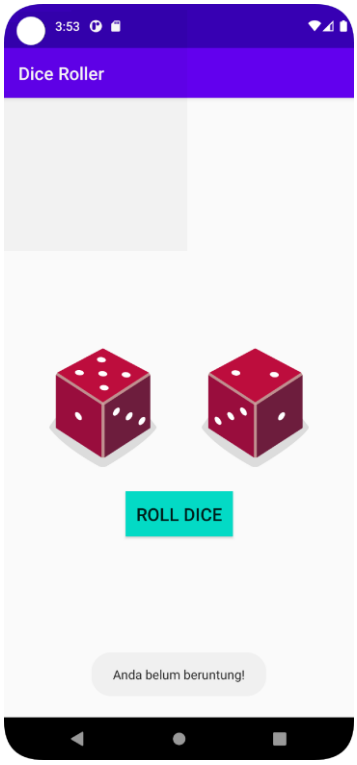
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



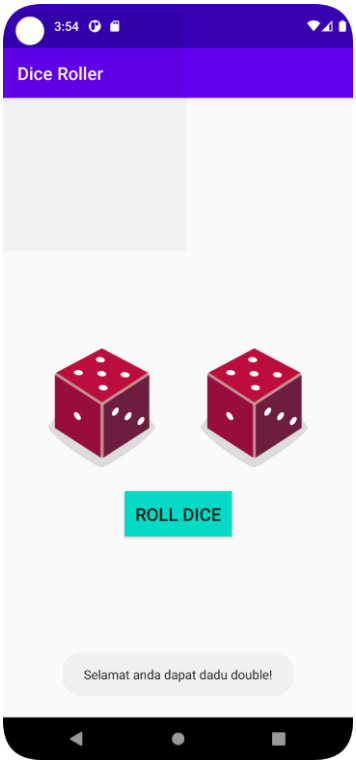
Gambar 1 Tampilan Awal Aplikasi

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



Gambar 2 Tampilan Dadu Setelah Di Roll

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



Gambar 3 Tampilan Roll Dadu Double

A. Source Code

1. MainActivity.kt

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

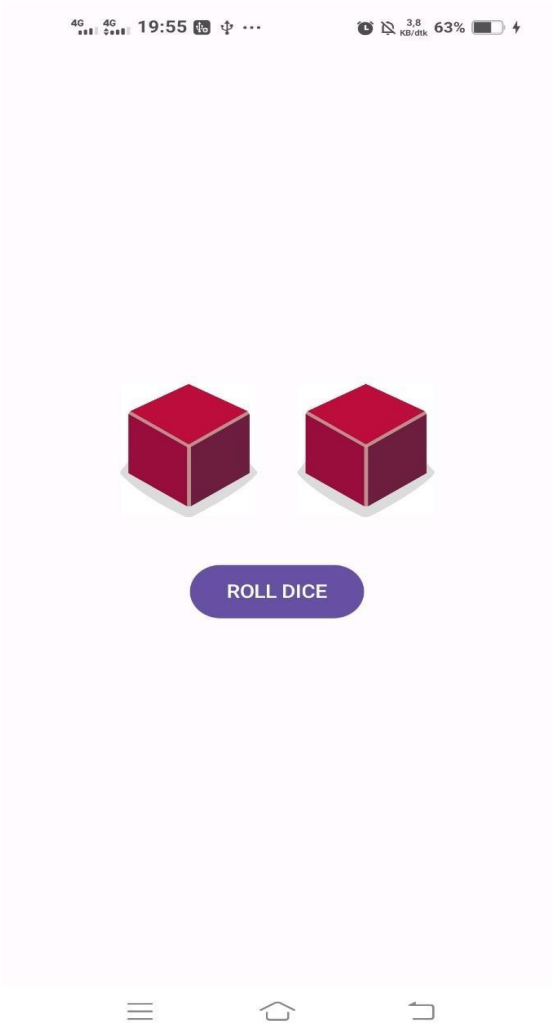
```
1   package com.example.diceroller
2
3
4   import android.widget.Toast
5   import androidx.compose.ui.platform.LocalContext
6   import android.os.Bundle
7   import androidx.activity.ComponentActivity
8   import androidx.activity.compose.setContent
9   import androidx.activity.enableEdgeToEdge
10  import androidx.compose.foundation.Image
11  import androidx.compose.foundation.layout.*
12  import androidx.compose.material3.*
13  import androidx.compose.runtime.*
14  import androidx.compose.ui.Alignment
15  import androidx.compose.ui.Modifier
16  import androidx.compose.ui.res.painterResource
17  import androidx.compose.ui.unit.dp
18  import androidx.compose.ui.unit.sp
19  import androidx.compose.ui.tooling.preview.Preview
20  import com.example.diceroller.ui.theme.DiceRollerTheme
```

```
21  import kotlin.random.Random

22  class MainActivity : ComponentActivity() {
23      override fun onCreate(savedInstanceState: Bundle?) {
24          super.onCreate(savedInstanceState)
25          enableEdgeToEdge()
26          setContent {
27              DiceRollerTheme {
28                  Scaffold(modifier = Modifier.fillMaxSize()) {
    innerPadding ->
29                      DiceGameScreen(
30                          modifier                                =
    Modifier.padding(innerPadding)
31                      )
32                  }
33              }
34          }
35      }
36  }
37
38  @Composable
39  fun DiceGameScreen(modifier: Modifier = Modifier) {
40      var dice1 by remember { mutableStateOf(0) }
41      var dice2 by remember { mutableStateOf(0) }
42      val context = LocalContext.current
43      var resultText by remember { mutableStateOf("") }
44
45      val getDiceImage = { value: Int ->
46          when (value) {
47              1 -> R.drawable.dice_1
48              2 -> R.drawable.dice_2
49              3 -> R.drawable.dice_3
50              4 -> R.drawable.dice_4
51              5 -> R.drawable.dice_5
52              6 -> R.drawable.dice_6
53              else -> R.drawable.empty_dice
54          }
55      }
56
57      Column(
58          modifier = modifier
59              .fillMaxSize()
60              .padding(24.dp),
61          horizontalAlignment = Alignment.CenterHorizontally,
62          verticalArrangement = Arrangement.Center
63      ) {
64          Row(horizontalArrangement                               =
65  Arrangement.spacedBy(16.dp)) {
66              Image(
67                  painter      =      painterResource(id      =
    getDiceImage(dice1)),
68                  contentDescription = "Dadu 1",
```

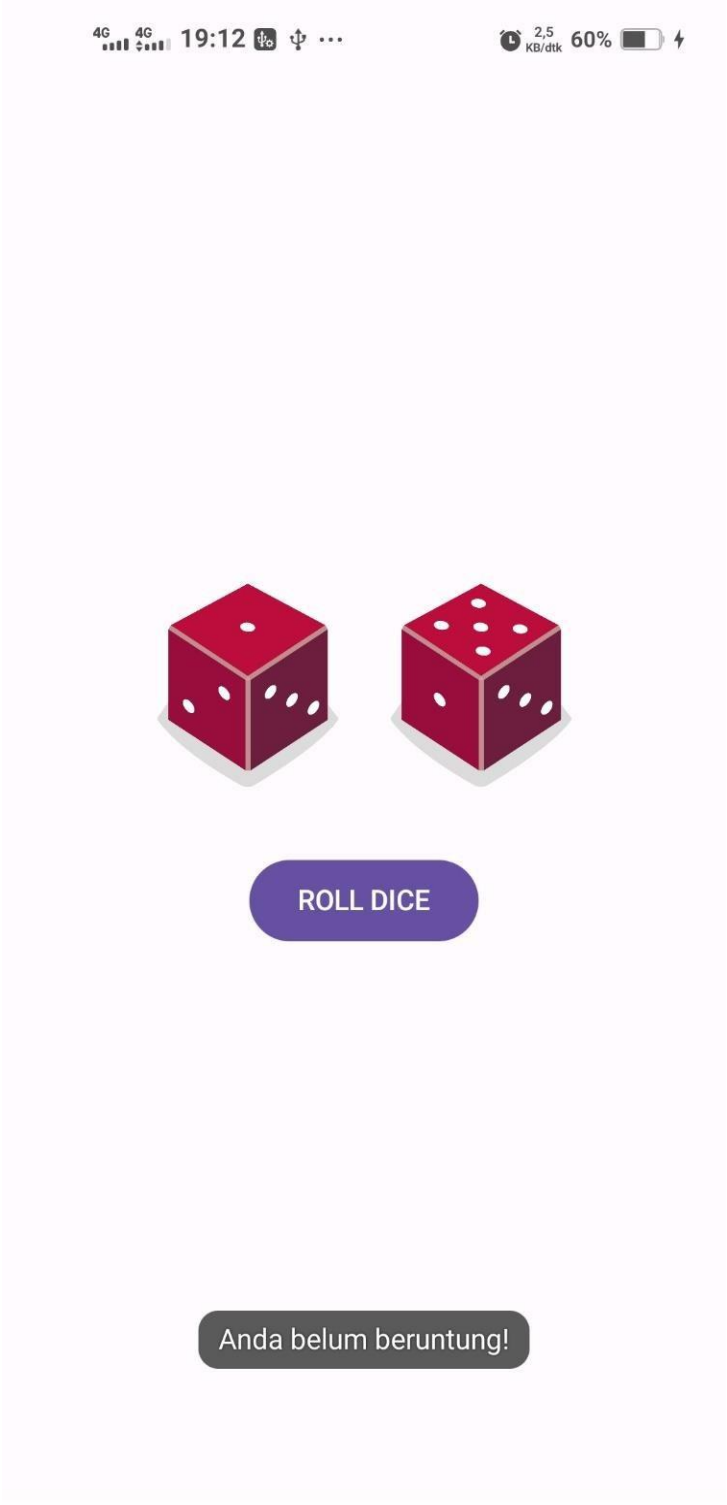
```
69                  modifier = Modifier.size(100.dp)
70              )
71              Image(
72                  painter      =      painterResource(id      =
73  getDiceImage(dice2)),
                    contentDescription = "Dadu 2",
74                  modifier = Modifier.size(100.dp)
75              )
76          }
77
78          Spacer(modifier = Modifier.height(32.dp))
79
80          Button(onClick = {
81              dice1 = Random.nextInt(1, 6)
82              dice2 = Random.nextInt(1, 6)
83              if (dice1 == dice2) {
84                  Toast.makeText(context,      "Selamat      anda
    mendapatkan dadu double!", Toast.LENGTH_SHORT).show()
85              } else {
86                  Toast.makeText(context,       "Anda       belum
    beruntung!", Toast.LENGTH_SHORT).show()
87              }
88          }) {
89              Text("ROLL DICE")
90          }
91
92          Spacer(modifier = Modifier.height(16.dp))
93
94          if (resultText.isNotEmpty()) {
95              Surface(
96                  modifier = Modifier
97                      .fillMaxWidth()
98                      .padding(top = 16.dp),
99                  shape = MaterialTheme.shapes.medium,
100                 tonalElevation = 2.dp,
101                 color                                          =
102 MaterialTheme.colorScheme.surfaceVariant
              ) {
103                 Box(modifier = Modifier.padding(12.dp)) {
104                     Text(
105                         text = resultText,
106                         fontSize = 16.sp,
107                         modifier                               =
    Modifier.align(Alignment.Center)
108                     )
109                 }
110             }
111         }
112     }
113 }
114
115 @Preview(showBackground = true)
```

```
@Composable
fun DiceGamePreview() {
    DiceRollerTheme {
        DiceGameScreen()
    }
}
```

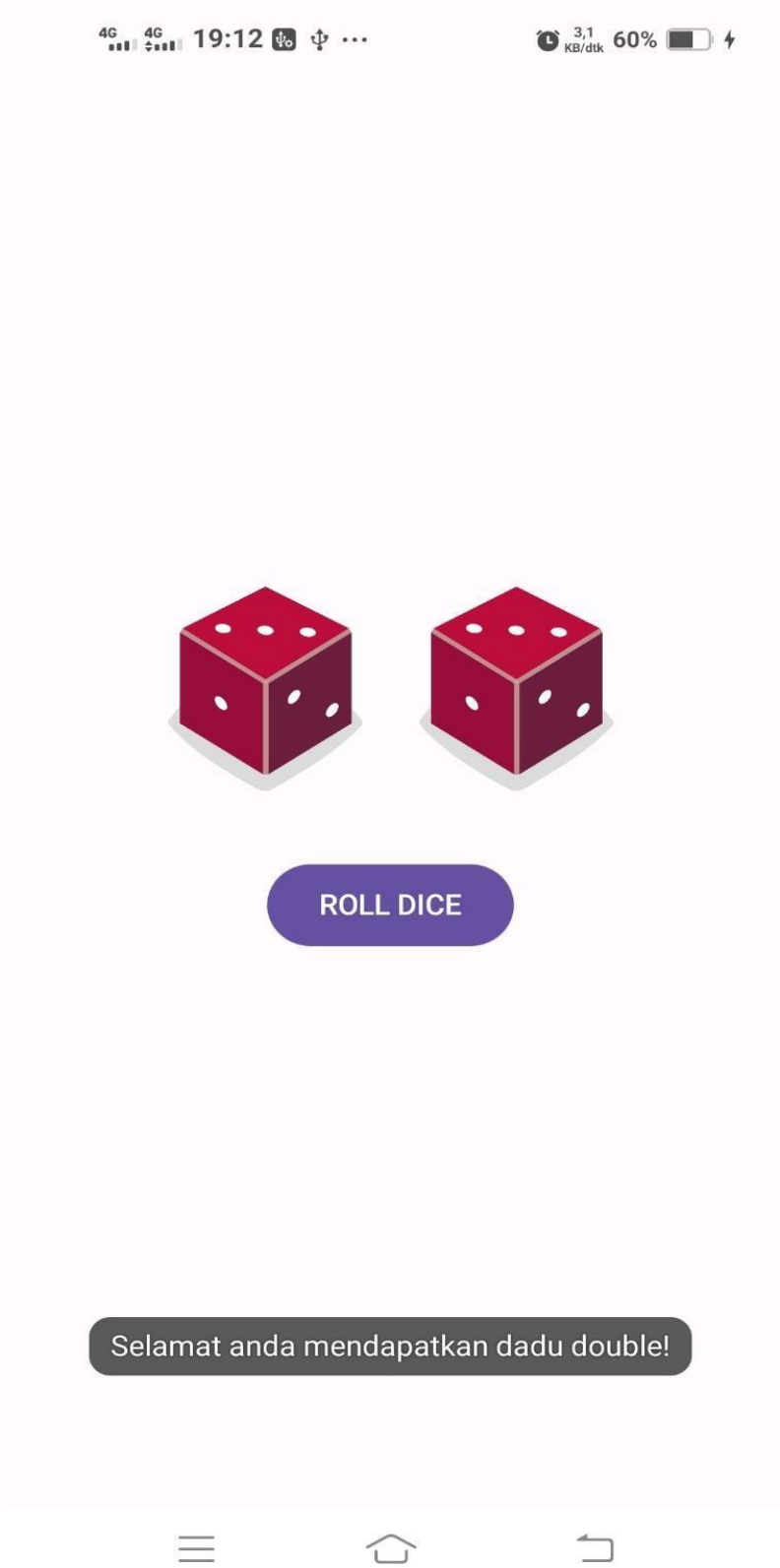
B. Output Program



Gambar 1. Tampilan Awal Aplikasi



Gambar 2. Tampilan Dadu Setelah di Roll, Jika Dadu Berbeda



Gambar 3. Tampilan Dadu Setelah di Roll, Jika Dadu Sama

C. Pembahasan

1. MainActivity.kt:

Pada praktikum ini dibuat sebuah aplikasi Android sederhana bernama Dice Roller menggunakan Jetpack Compose. Aplikasi ini berfungsi untuk melakukan simulasi lempar dua buah dadu secara acak. Proses utama aplikasi berada di dalam kelas `MainActivity`, di mana di dalam metode `onCreate`, dipanggil fungsi `setContent` untuk menampilkan UI dengan tema `DiceRollerTheme`. UI utama dibuat dalam fungsi `DiceGameScreen`.

Di dalam `DiceGameScreen`, terdapat dua buah variabel `dice1` dan `dice2` yang masing-masing menyimpan angka hasil lemparan dadu. Variabel tersebut dibuat menggunakan `remember` dan `mutableStateOf` sehingga perubahan nilainya akan langsung memicu perubahan tampilan pada layar. Selain itu, terdapat juga variabel `resultText` untuk menyimpan hasil teks yang ingin ditampilkan (meskipun pada implementasi saat ini belum digunakan secara optimal).

Untuk menampilkan gambar dadu sesuai angka yang dihasilkan, dibuat sebuah fungsi `getDiceImage`. Fungsi ini menerima angka dari 1 hingga 6 dan mengembalikan resource gambar dadu yang sesuai. Jika nilai dadu belum diatur (default 0), maka akan ditampilkan gambar dadu kosong.

Tata letak antarmuka dibangun menggunakan `Column`, sehingga elemen UI tersusun secara vertikal. Gambar dua buah dadu ditempatkan dalam `Row` agar tampil secara horizontal dan diberi jarak antar gambar. Sebuah tombol dengan teks "ROLL DICE" diletakkan di bawah gambar. Ketika tombol ditekan, aplikasi akan mengacak nilai `dice1` dan `dice2` dengan `Random.nextInt(1, 6)`, kemudian membandingkan kedua nilai tersebut. Jika hasilnya sama, maka akan ditampilkan pesan Toast "Selamat anda mendapatkan dadu double!". Jika berbeda, akan ditampilkan Toast "Anda belum beruntung!".

Selain itu, disiapkan juga sebuah `Surface` yang seharusnya berfungsi untuk menampilkan teks hasil di bawah tombol. Namun karena `resultText` tidak diubah dalam aksi tombol, bagian ini belum berfungsi secara penuh. Akhirnya, dibuat fungsi `DiceGamePreview` untuk mempermudah melihat pratinjau tampilan aplikasimelalui fitur preview di Android Studio tanpa harus menjalankan aplikasi di emulator atau perangkat fisik.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/annacoded/Pemrograman-Mobile

MODUL 2 : ANDROID LAYOUT

SOAL 1

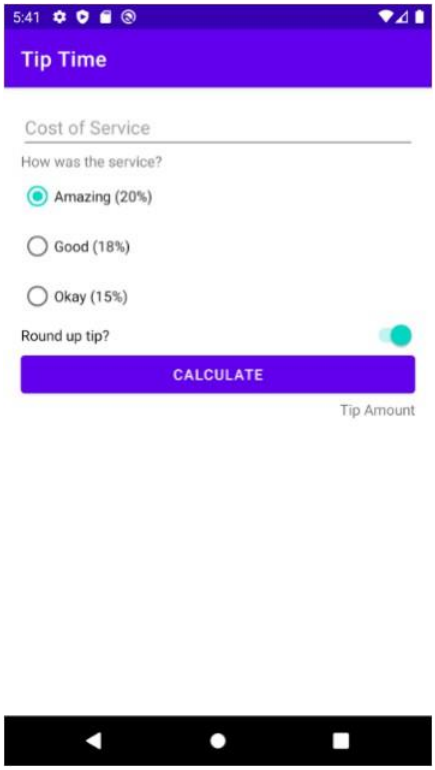
Buatlah sebuah aplikasi kalkulator tip yang dirancang untuk membantu pengguna menghitung tip yang sesuai berdasarkan total biaya layanan yang mereka terima. Fitur-fitur yang diharapkan dalam aplikasi ini mencakup:

1. Input Biaya Layanan: Pengguna dapat memasukkan total biaya layanan yang diterima dalam bentuk nominal.

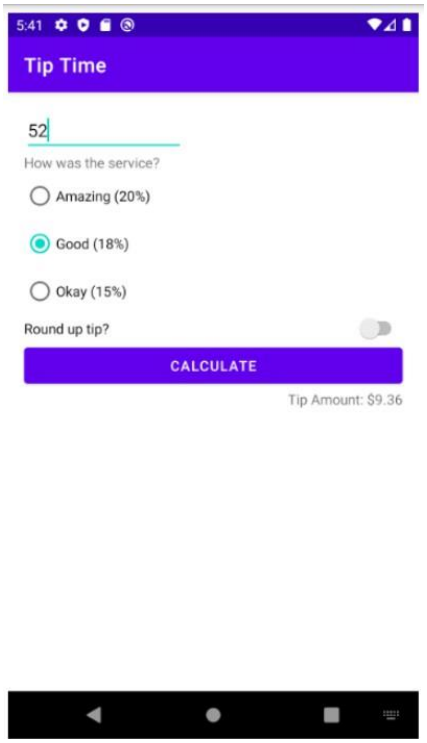
2. Pilihan Persentase Tip: Pengguna dapat memilih persentase tip yang diinginkan dari opsi yang disediakan, yaitu 15%, 18%, dan 20%.

3. Pengaturan Pembulatan Tip: Pengguna dapat memilih untuk membulatkan tip ke angka yang lebih tinggi.

4. Tampilan Hasil: Aplikasi akan menampilkan jumlah tip yang harus dibayar secara langsung setelah pengguna memberikan input.



Gambar 1 Tampilan Awal Aplikasi



Gambar 2 Tampilan Aplikasi Setelah Dijalankan

A. Source Code

1. MainActivity.kt

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

```
1   package com.example.tipcalc
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import android.widget.Toast
6   import com.example.tipcalc.databinding.ActivityMainBinding
7
8
9   class MainActivity : ComponentActivity() {
10
11      lateinit var binding: ActivityMainBinding
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          binding = ActivityMainBinding.inflate(layoutInflater)
16          setContentView(binding.root)
17          binding.calculate.setOnClickListener {
18              hitung()
19          }
20      }
21      private fun hitung(){
22          val                          cost                          =
    binding.costOfService.text.toString().toDouble()
23          val tip = binding.tipOptions.checkedRadioButtonId
24          val tipPercent = when (tip) {
25              R.id.option_twenty -> 0.20
26              R.id.option_eighteen -> 0.18
27              else -> 0.15
28          }
29
30          var total = tipPercent * cost
31          val roundUp = binding.roundUp.isChecked
32          if(roundUp){
33              total = kotlin.math.ceil(total)
            }
34          if(cost <= 0){
35              Toast.makeText(this, "Masukkan angka yang valid",
    Toast.LENGTH_SHORT).show()
36          }else{
37              binding.total.text = total.toString()
38          }
39      }
40  }
```

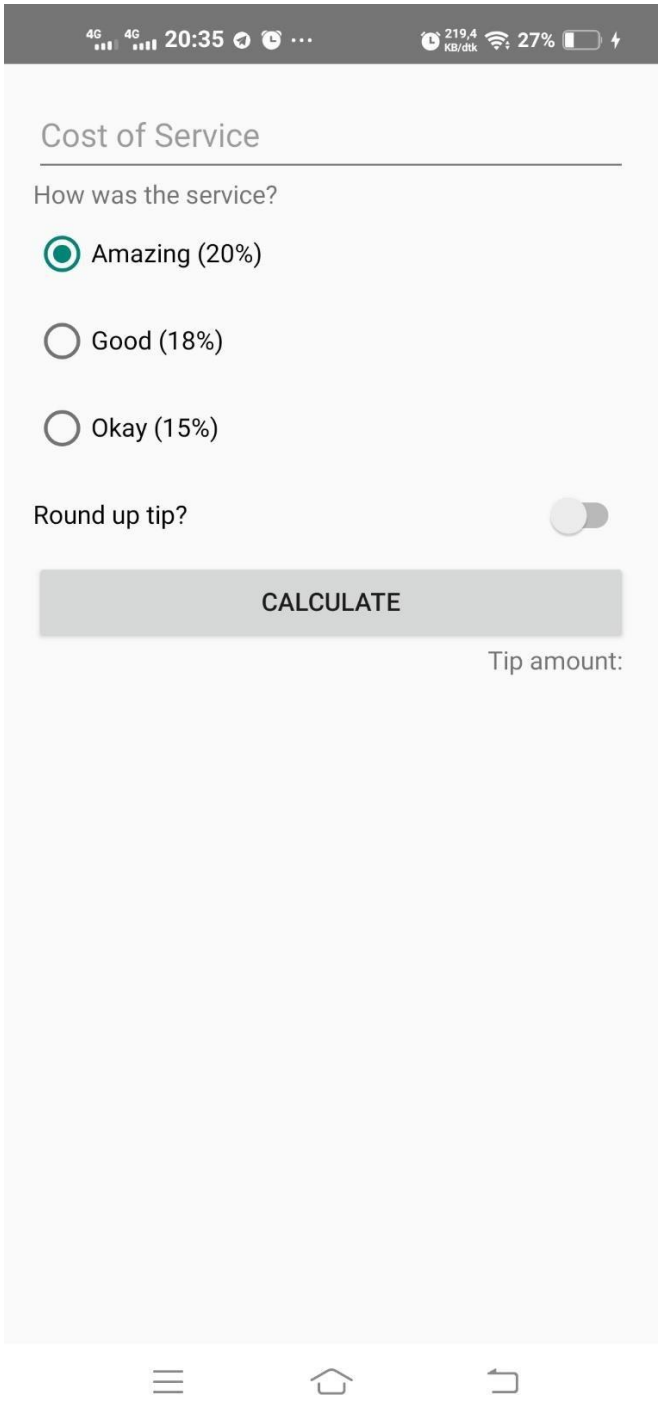
2. Activity_main.xml

Tabel 3. Source Code Jawaban Soal 1Modul 2 Activity_main.xml

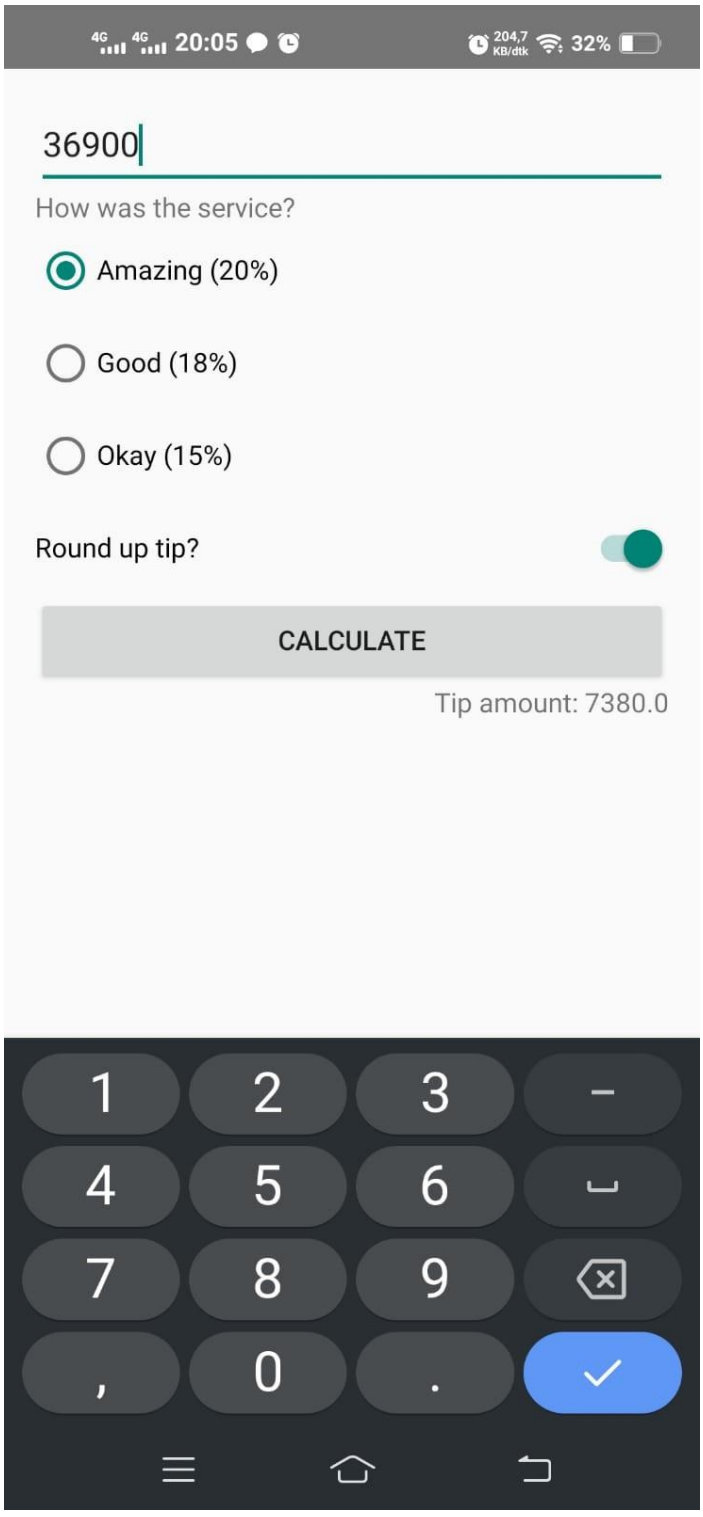
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/cost_of_service"
        android:hint="Cost of Service"
        android:inputType="number"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        />
    <TextView
        android:id="@+id/service_question"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/cost_of_service"
        android:text="How was the service?"
        />
    <RadioGroup
        android:id="@+id/tip_options"
        app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toBottomOf="@id/service_question"
        android:checkedButton="@id/option_twenty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">

        <RadioButton
            android:id="@+id/option_twenty"
            android:layout_width="wrap_content"
            android:layout_height="48dp"
            android:text="Amazing (20%)"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            />
        <RadioButton
            android:id="@+id/option_eighteen"
            android:layout_width="wrap_content"
```

```
            android:layout_height="48dp"
            android:text="Good (18%)"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            />
        <RadioButton
            android:id="@+id/option_fifteen"
            android:layout_width="wrap_content"
            android:layout_height="48dp"
            android:text="Okay (15%)"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            />
    </RadioGroup>
    <Switch
        android:id="@+id/round_up"
        android:text="Round up tip?"
        app:layout_constraintStart_toStartOf="parent"
       app:layout_constraintTop_toBottomOf="@id/tip_options"
        android:layout_width="match_parent"
        android:layout_height="48dp"
        />
    <Button
        android:id="@+id/calculate"
        android:text="Calculate"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/round_up"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView
        android:id="@+id/text"
        android:text="Tip amount: "
        app:layout_constraintRight_toLeftOf="@id/total"
        app:layout_constraintTop_toBottomOf="@+id/calculate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
    <TextView
        android:id="@+id/total"
        android:text=""
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/calculate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

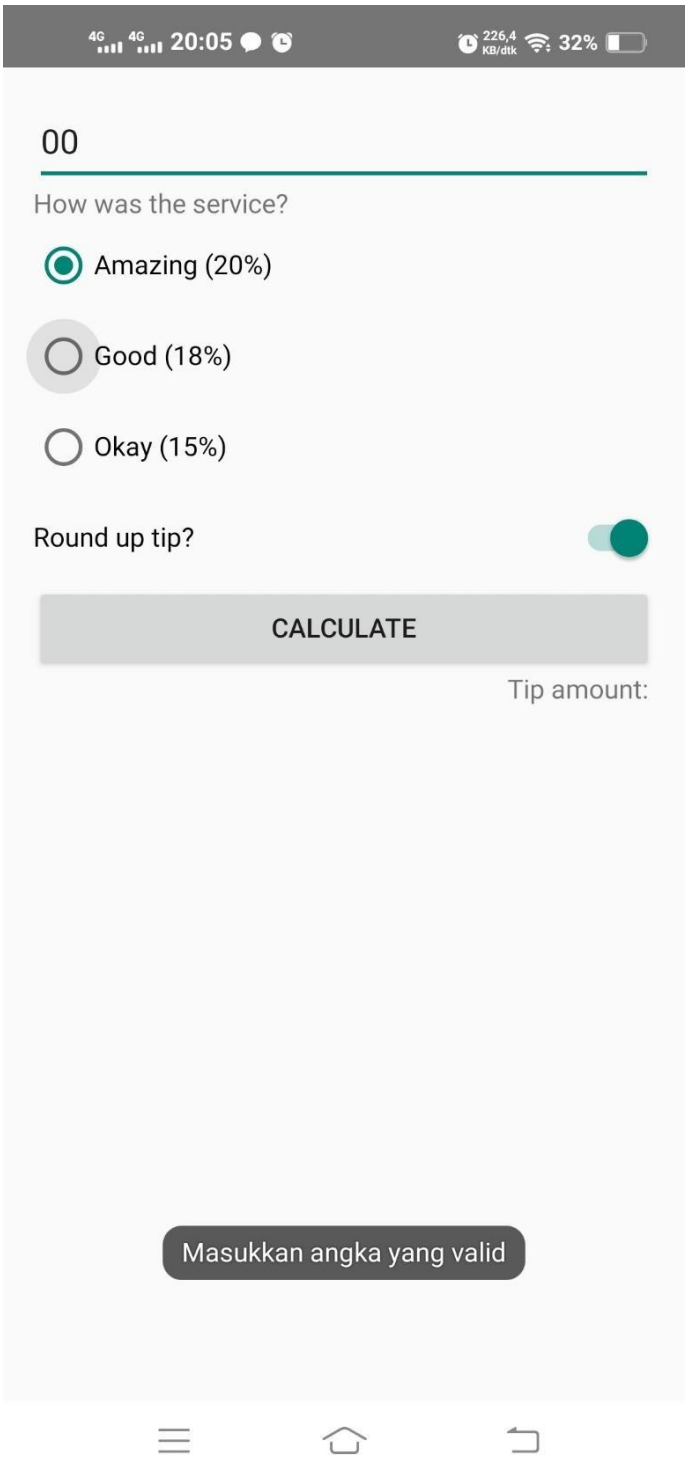
B. Output Program



Gambar 4. Tampilan Awal Aplikasi



Gambar 5. Tampilan Aplikasi Setelah Dijalankan



Gambar 6. Tampilan Saat Memasukkan Angka yang Tidak Valid

C. Pembahasan

1. MainActivity.kt:

Kode ini adalah program Android sederhana yang fungsinya untuk menghitung tip atau uang jasa tambahan dari suatu layanan. Program ini ditulis menggunakan Kotlin dan untuk akases elemen-elemen di layout-nya pakai View Binding, agar lebih mudah dan tidak perlu pakai `findViewById()`. Saat aplikasi dibuka, bagian `onCreate()` akan jalan duluan. Di situ kita nge-setup tampilan dengan `ActivityMainBinding`, Lalu, mengatur agar tombol "calculate" bisa merespon klik dari pengguna. Kalau tombolnya diklik, dia akan memanggil fungsi `hitung()`.

Di dalam fungsi `hitung()`, pertama-tama kita ambil nilai dari kolom input yang isinya biaya layanan, lalu diubah jadi `Double` (angka desimal). Lalu, kita cek juga radio button mana yang dipilih buat nunjukin persentase tip — bisa 20%, 18%, atau 15%.

Setelah itu, kita hitung jumlah tip-nya dari persentase dikali biaya. Kalau kotak "round up" dicentang, maka hasil tip-nya kita bulatkan ke atas (pakai `ceil()`).

Tapi, sebelum menampilkan hasilnya, kita cek dulu apakah biaya yang dimasukin itu lebih kecil dari atau sama dengan nol. Kalau iya, muncul `Toast` atau pesan peringatan yang ngasih tahu kalau input-nya nggak valid. Kalau angkanya valid, bar kita tampilkan hasil tip-nya ke tampilan aplikasi.

2. Activity_main.xml:

Layout XML ini adalah tampilan utama untuk aplikasi kalkulator tip yang dibuat menggunakan `ConstraintLayout`, yang memudahkan pengaturan posisi tiap elemen secara fleksibel dan saling terhubung. Di bagian paling atas, ada `EditText` dengan ID `cost_of_service` yang digunakan untuk memasukkan biaya layanan. Pengguna hanya bisa memasukkan angka, dan ada hint yang bilang "Cost of Service" supaya pengguna tahu apa yang harus diisi. Setelah itu, ada `TextView` yang bertuliskan "How was the service?", yang berfungsi untuk memberikan petunjuk kepada pengguna sebelum memilih tingkat kepuasan mereka. Di bawahnya, ada `RadioGroup` yang berisi tiga `RadioButton` yang menunjukkan pilihan persentase tip, yaitu "Amazing (20%)", "Good (18%)", dan "Okay

(15%)". Pilihan ini menentukan berapa persen tip yang akan dihitung berdasarkan biaya yang dimasukkan. Kemudian ada sebuah `Switch` dengan ID `round_up`, yang memberi opsi kepada pengguna apakah ingin membulatkan jumlah tip ke atas atau tidak. Setelah itu, ada tombol `Button` dengan teks "Calculate", yang nantinya akan mengkalkulasi tip ketika diklik oleh pengguna. Hasil perhitungan tip akan ditampilkan di dua `TextView`: yang pertama menunjukkan label "Tip amount:", dan yang kedua akan menampilkan hasil tip yang sudah dihitung. Semua elemen ini diatur dengan rapi supaya mudah digunakan dan dilihat oleh pengguna.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/annacoded/Pemrograman-Mobile

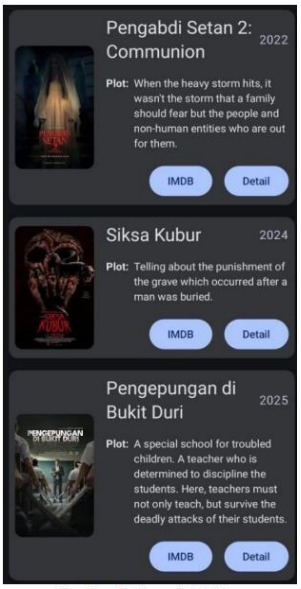
.

MODUL 3 :BUILD SCROLLABLE LIST

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

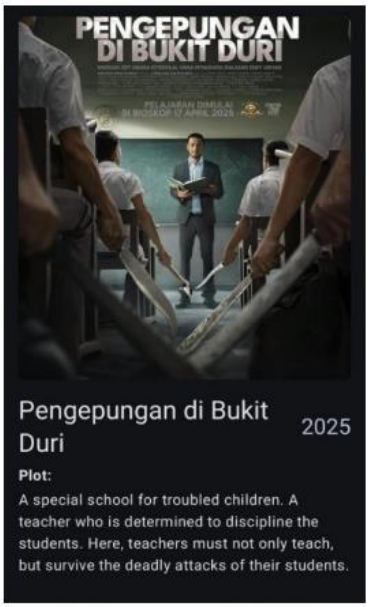
1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4.Terdapat 2 button dalam list, dengan fungsi berikut:
    a. Button pertama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

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



Gambar 1. Contoh UI List

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



Gambar 2. Contoh UI Detail

A. Source Code

1. MainActivity.kt

Tabel 4. Source Code Jawaban Soal 1 Modul 3 MainActivity.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import androidx.activity.compose.setContent
5  import androidx.activity.enableEdgeToEdge
6  import androidx.compose.foundation.layout.fillMaxSize
7  import androidx.compose.foundation.layout.padding
8  import androidx.compose.material3.Scaffold
9  import androidx.compose.material3.Text
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.tooling.preview.Preview
13 import androidx.appcompat.app.AppCompatActivity
14 import androidx.navigation.fragment.NavHostFragment
15 import com.example.dreamstravel.databinding.ActivityMainBinding
16 import com.example.dreamstravel.ui.theme.DreamsTravelTheme
17
18 class MainActivity : AppCompatActivity() {
19
20     private lateinit var binding: ActivityMainBinding  // ←
   Binding object
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         val navHostFragment =
   supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
   as NavHostFragment
29         val navController = navHostFragment.navController
30     }
31 }
```

2. HomeFragment.kt

Tabel 5. Source Code Jawaban Soal 1 Modul 3 HomeFragment.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import android.view.LayoutInflater
5  import android.view.View
6  import android.view.ViewGroup
7  import androidx.fragment.app.Fragment
8  import androidx.fragment.app.viewModels
9  import androidx.navigation.fragment.findNavController
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import com.example.dreamstravel.databinding.FragmentHomeBinding
12
13 class HomeFragment : Fragment() {
14     private var _binding: FragmentHomeBinding? = null
15     private val binding get() = _binding!!
16
17     private val viewModel: NegaraViewModel by viewModels()
18
19     override fun onCreateView(inflater: LayoutInflater,
   container: ViewGroup?, savedInstanceState: Bundle?): View? {
20         _binding = FragmentHomeBinding.inflate(inflater,
   container, false)
21         return binding.root
22     }
23
24     override fun onViewCreated(view: View, savedInstanceState:
   Bundle?) {
25         val adapter = NegaraAdapter(viewModel.loadNegara(),
   findNavController())
26         binding.rvNegara.layoutManager =
   LinearLayoutManager(requireContext())
27         binding.rvNegara.adapter = adapter
28     }
29
30     override fun onDestroyView() {
31         super.onDestroyView()
32         _binding = null
33     }
34 }
```

3. DetailFragment.kt

Tabel 6. Source Code Jawaban Soal 1 Modul 3 DetailFragment.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import androidx.fragment.app.Fragment
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import
   com.example.dreamstravel.databinding.FragmentDetailBinding
9
10 class DetailFragment : Fragment() {
11
12     private var _binding: FragmentDetailBinding? = null
13     private val binding get() = _binding!!
14
15     override fun onCreateView(
16         inflater: LayoutInflater,
17         container: ViewGroup?,
18         savedInstanceState: Bundle?
19     ): View? {
20         _binding = FragmentDetailBinding.inflate(inflater,
   container, false)
21         return binding.root
22     }
23
24     override fun onViewCreated(view: View, savedInstanceState:
   Bundle?) {
25         val namaNegara = arguments?.getString("namaNegara") ?:
   "Tidak ada nama"
26         val alasan = arguments?.getString("alasan") ?: "Tidak
   ada tahun"
27         val gambar = arguments?.getInt("gambarResId") ?:
   R.drawable.japan
28
29         binding.imgCountry.setImageResource(gambar)
30         binding.tvCountryName.text = namaNegara
31         binding.tvCountryReason.text = alasan
32     }
33
34     override fun onDestroyView() {
35         super.onDestroyView()
36         _binding = null
37     }
38 }
```

4. NegaraAdapter.kt

Tabel 7. Source Code Jawaban Soal 1 Modul 3 NegaraAdapter.kt

```
1  package com.example.dreamstravel
2
3  import android.content.Intent
4  import androidx.core.net.toUri
5  import android.view.LayoutInflater
6  import android.view.ViewGroup
7  import androidx.navigation.NavController
8  import androidx.recyclerview.widget.RecyclerView
9  import com.example.dreamstravel.databinding.ItemNegaraBinding
10
11 class NegaraAdapter(
12     private val listNegara: List<NegaraModel>,
13     private val navController: NavController
14 ) : RecyclerView.Adapter<NegaraAdapter.NegaraViewHolder>() {
15
16     inner class NegaraViewHolder(val binding: ItemNegaraBinding)
   : RecyclerView.ViewHolder(binding.root)
17
18     override fun onCreateViewHolder(parent: ViewGroup, viewType:
   Int): NegaraViewHolder {
19         val binding =
   ItemNegaraBinding.inflate(LayoutInflater.from(parent.context),
   parent, false)
20         return NegaraViewHolder(binding)
21     }
22
23     override fun onBindViewHolder(holder: NegaraViewHolder,
   position: Int) {
24         val negara = listNegara[position]
25         holder.binding.tvItemName.text = negara.nama
26
27 holder.binding.imgItemPhoto.setImageResource(negara.gambarResId)
28         holder.binding.btnWiki.setOnClickListener{
29             var url = negara.wikiUrl
30             val intent = Intent(Intent.ACTION_VIEW, url.toUri())
31             it.context.startActivity(intent)
32         }
33
34         holder.binding.buttonDetail.setOnClickListener {
35             val bundle = android.os.Bundle().apply {
36                 putInt("gambarResId", negara.gambarResId)
37                 putString("namaNegara", negara.nama)
38                 putString("alasan", negara.alasan)
39             }
40             navController.navigate(R.id.detailFragment, bundle)
```

```
41          }
42      }
43      override fun getItemCount() = listNegara.size
44  }
```

5. NegaraModel.kt

Tabel 8. Source Code Jawaban Soal 1 Modul 3 NegaraModelFragment.kt

```
1   package com.example.dreamstravel
2
3
4   data class NegaraModel(
5       val nama: String,
6       val tahun: String,
7       val alasan: String,
8       val gambarResId: Int,
9       val wikiUrl: String
10  )
```

6. NegaraViewModel.kt

Tabel 9. Source Code Jawaban Soal 1 Modul 3 NegaraViewModel.kt

```
1  package com.example.dreamstravel
2  import android.app.Application
3  import androidx.lifecycle.AndroidViewModel
4  class NegaraViewModel(application: Application) :
   AndroidViewModel(application) {
5      private val context =
   getApplication<Application>().applicationContext
6
7
8      fun loadNegara(): List<NegaraModel> {
9          val namaNegara =
   context.resources.getStringArray(R.array.list_negara)
10         val tahun =
   context.resources.getStringArray(R.array.list_tahun)
11         val alasan =
   context.resources.getStringArray(R.array.list_alasan)
12         val link =
   context.resources.getStringArray(R.array.list_link)
13         val gambar = listOf(
14             R.drawable.japan,
15             R.drawable.swiss,
16             R.drawable.korsel,
17             R.drawable.arabsaudi,
18             R.drawable.france
19         )
20
21         return namaNegara.indices.map { i ->
22             NegaraModel(
23                 namaNegara[i],
24                 tahun[i],
25                 alasan[i],
26                 gambar[i],
27                 link[i]
28             )
29         }
30     }
31 }
```

7. Activity_main.xml

Tabel 10. Source Code Jawaban Soal 1 Modul 3 Activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Selamat datang di DreamsTravel!"
13         android:textSize="18sp"
14         android:layout_gravity="center_horizontal"
15         android:layout_margin="16dp"/>
16
17     <androidx.fragment.app.FragmentContainerView
18         android:id="@+id/nav_host_fragment"
19
20 android:name="androidx.navigation.fragment.NavHostFragment"
21         android:layout_width="match_parent"
22         android:layout_height="match_parent"
23         app:defaultNavHost="true"
24         app:navGraph="@navigation/nav_graph" />
25 </LinearLayout>
```

8. Fragment_detail.xml

Tabel 11. Source Code Jawaban Soal 1 Modul 3 Fragment_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".DetailFragment">
8
9      <ImageView
10         android:id="@+id/imgCountry"
11         android:layout_width="200dp"
```

```
        android:layout_height="200dp"
        android:layout_marginTop="32dp"
        android:layout_centerHorizontal="true"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/tvCountryName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"

app:layout_constraintTop_toBottomOf="@id/imgCountry"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:text="Country Name"/>

    <TextView
        android:id="@+id/tvCountryReason"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

app:layout_constraintTop_toBottomOf="@id/tvCountryName"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        android:text="Reason"/>
    </androidx.constraintlayout.widget.ConstraintLayout>
```

9. Fragment_home.xml

Tabel 12. Source Code Jawaban Soal 1 Modul 3 Fragment_home.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:tools="http://schemas.android.com/tools">
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvNegara"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:clipToPadding="false"
        android:padding="8dp"
```

```
13          android:contentDescription="cute"
14          app:layout_constraintTop_toTopOf="parent"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintEnd_toEndOf="parent"
18          tools:listitem="@layout/item_negara" />
19      </androidx.constraintlayout.widget.ConstraintLayout>
```

10. Item_negara.xml

Tabel 13. Source Code Jawaban Soal 1 Modul 3 Item_negara.xml

```
1   <?xml version=”1.0” encoding=”utf-8”?>
2   <15ndroid.cardview.widget.CardView
    xmlns:android=”http://schemas.android.com/apk/res/android”
3       xmlns:card_view=”http://schemas.android.com/apk/res-
    auto”
4       xmlns:tools=”http://schemas.android.com/tools”
5       xmlns:app=”http://schemas.android.com/apk/res-auto”
6       android:layout_width=”match_parent”
7       android:layout_height=”wrap_content”
8       android:layout_margin=”@dimen/card_margin”
9       app:cardCornerRadius=”@dimen/card_radius”
10      app:cardElevation=”4dp”>
11
12      <15ndroid.constraintlayout.widget.ConstraintLayout
13          android:layout_width=”match_parent”
14          android:layout_height=”wrap_content”
15          android:padding=”@dimen/card_padding”>
16
17
    <com.google.android.material.imageview.ShapeableImageView
18              android:id=”@+id/img_item_photo”
19              android:layout_width=”80dp”
20              android:layout_height=”100dp”
21              android:scaleType=”centerCrop”
22              app:layout_constraintStart_toStartOf=”parent”
23              app:layout_constraintTop_toTopOf=”parent”
24
    app:layout_constraintBottom_toBottomOf=”parent”
25
```

```
            android:contentDescription="@string/img_desc"
/>

        <TextView
            android:id="@+id/tv_item_name"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:textSize="18sp"
            android:textStyle="bold"
            android:textColor="@android:color/black"

app:layout_constraintStart_toEndOf="@id/img_item_photo"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintHorizontal_bias="0"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            tools:text="Jepang" />

        <Button
            android:id="@+id/button_detail"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btn_detail"
            android:minHeight="48dp"
            android:minWidth="48dp"

android:paddingHorizontal="@dimen/button_padding"

app:layout_constraintTop_toBottomOf="@id/tv_item_name"

app:layout_constraintStart_toStartOf="@id/tv_item_name"

app:layout_constraintBottom_toBottomOf="parent" />

        <Button
            android:id="@+id/btn_wiki"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btn_wiki"
            android:minHeight="48dp"
            android:minWidth="48dp"

android:paddingHorizontal="@dimen/button_padding"

app:layout_constraintTop_toBottomOf="@id/tv_item_name"

app:layout_constraintStart_toEndOf="@id/button_detail"
```

```
64
   app:layout_constraintBottom_toBottomOf=”parent”
65              android:layout_marginStart=”8dp” />
66
67      </android.constraintlayout.widget.ConstraintLayout>

68  </androidx.cardview.widget.CardView>
```

11. nav_graph.xml

Tabel 14. Source Code Jawaban Soal 1 Modul 3 nav_graph.xml

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

12. strings.xml

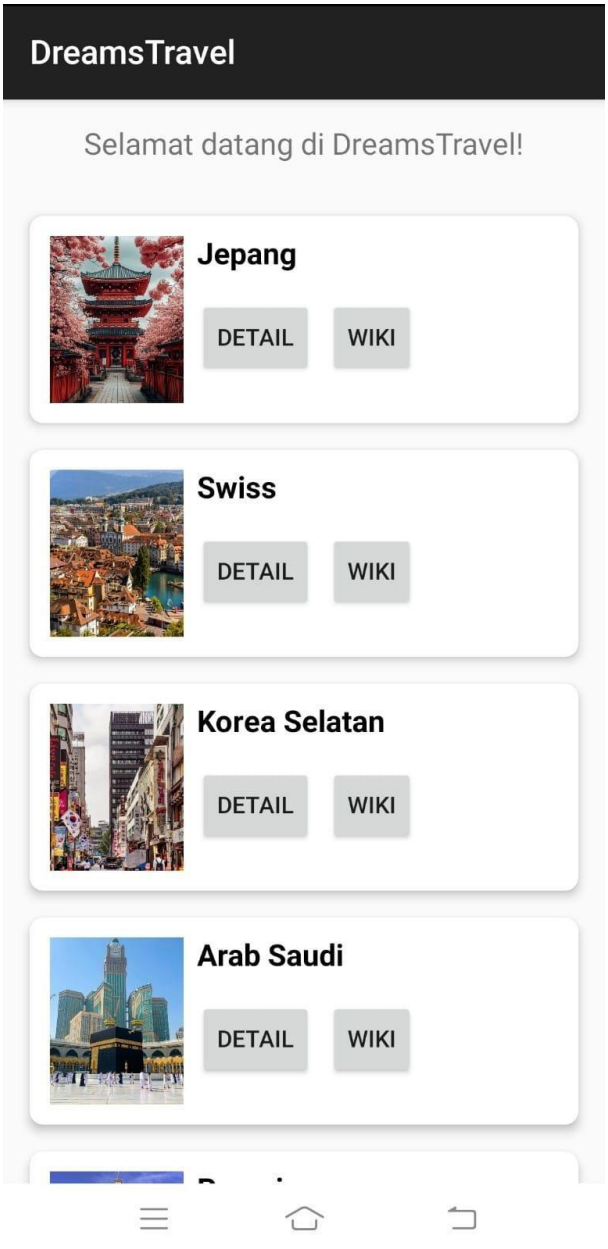
Tabel 15. Source Code Jawaban Soal 1 Modul 3 string.xml

```
1   <resources>
2       <string name="app_name">DreamsTravel</string>
```

```
3       <string name="btn_detail">Detail</string>
4       <string name="btn_wiki">Wiki</string>
5       <string name="img_desc">Gambar negara impian</string>
6       <string-array name="list_negara">
7           <item>Jepang</item>
8           <item>Swiss</item>
9           <item>Korea Selatan</item>
10          <item>Arab Saudi</item>
11          <item>Prancis</item>
12      </string-array>
13      <string-array name="list_tahun">
14          <item>2025</item>
15          <item>2026</item>
16          <item>2027</item>
17          <item>2028</item>
18          <item>2029</item>
19      </string-array>
20      <string-array name="data_gambar">
21          <item>@drawable/japan</item>
22          <item>@drawable/swiss</item>
23          <item>@drawable/korsel</item>
24          <item>@drawable/arabsaudi</item>
25          <item>@drawable/france</item>
26      </string-array>
27      <string-array name="list_alasan">
28          <item>Saya suka negara jepang karena budaya dan
    ingin ketemu Kento Yamazaki</item>
29          <item>Negara Swiss itu pemandangan alamnya bagus
    banget, kayak alam mimpi!</item>
30          <item>Saya suka K-Pop dan K-Dramanya juga semua
    sangat seru, alasan saya ingin kesini karena ingin bertemu
    Enhypen</item>
31          <item>Karena saya beragama islam, saya ingin
    melaksanakan ibadah haji</item>
32          <item>Saya juga ingin ke Prancis, karena katanya
    paris ini kota yang romantis</item>
33      </string-array>
34
35      <string-array name="list_link">
36          <item>https://en.wikipedia.org/wiki/Japan</item>
37
    <item>https://en.wikipedia.org/wiki/Switzerland</item>
38
    <item>https://id.wikipedia.org/wiki/Korea_Selatan</item>
39
    <item>https://id.wikipedia.org/wiki/Arab_Saudi</item>
40          <item>https://id.wikipedia.org/wiki/Prancis</item>
```

```
41      </string-array>
42
43  </resources>
```

B. Output Program



Gambar 7. Screenshot Halaman Utama

Gambar 8. Screenshot Halaman Detail

Gambar 9. Screenshot Halaman Link Wikipedia

C. Pembahasan

1. MainActivity.kt:

Kode di atas adalah isi dari file MainActivity.kt dalam proyek Android yang menggunakan Jetpack Navigation Component dan View Binding. Kelas MainActivity ini merupakan entry point atau layar utama saat aplikasi pertama kali dijalankan. Kode ini menggunakan pendekatan berbasis Activity tradisional (bukan Compose) meskipun ada sedikit bagian Compose yang belum digunakan secara aktif.

Pertama-tama, ada properti binding yang dibuat dengan tipe ActivityMainBinding. Ini adalah fitur View Binding yang memudahkan kita mengakses elemen-elemen XML tanpa perlu pakai `findViewById()`. Di dalam `fungsi onCreate()`, yaitu fungsi yang pertama kali dijalankan saat activity dibuat, kita menginisialisasi objek binding dengan cara meng-inflate layout dari activity_main.xml, lalu menampilkannya dengan `setContentView(binding.root)`.

Selanjutnya, kita mengambil fragment navigasi (komponen yang mengatur perpindahan antar layar/halaman di aplikasi) yang ada di dalam activity_main.xml melalui ID `nav_host_fragment`. Fragment ini kemudian dikonversi ke tipe `NavHostFragment` agar bisa kita akses `navController-nya`, yaitu komponen yang mengatur logika navigasi seperti berpindah halaman. Meskipun tidak ada aksi lanjut di kode ini, biasanya navController ini digunakan untuk mengatur arah navigasi berdasarkan interaksi pengguna.

2. HomeFragment.kt

Kode di atas adalah isi dari `HomeFragment.kt`, yaitu salah satu tampilan (fragment) di dalam aplikasi Android yang menampilkan daftar negara menggunakan `RecyclerView`. Fragment ini memakai View Binding dan juga `ViewModel` untuk mengambil data.

Pertama-tama, ada properti _binding yang digunakan untuk mengakses elemen-elemen layout dari file `fragment_home.xml` tanpa harus pakai `findViewById()`. Karena lifecycle fragment bisa berubah-ubah, binding ini di-null-kan kembali di `onDestroyView()` untuk mencegah kebocoran memori.

Kemudian, ada `viewModel` bertipe `NegaraViewModel` yang digunakan untuk mengambil data negara. `viewModels()` adalah cara instan untuk mendapatkan instance ViewModel yang sesuai dengan fragment ini.

Fungsi `onCreateView()` akan dipanggil saat fragment akan membuat tampilannya. Di sini, layout dari fragment di-inflate dan dikembalikan. Setelah tampilan dibuat, fungsi `onViewCreated()` akan dipanggil, dan di sinilah logika utama ditaruh. Dalam hal ini, dibuatlah sebuah adapter NegaraAdapter, yang isinya diisi dari data yang diambil lewat `viewModel.loadNegara()`. Adapter ini juga diberi `findNavController()` agar bisa mengatur navigasi saat item di-klik. RecyclerView kemudian disetel dengan LinearLayoutManager (agar tampilannya ke bawah seperti daftar biasa) dan dihubungkan dengan adapter.

3. DetailFragment.kt

Kode di atas adalah isi dari `DetailFragment.kt`, yaitu salah satu tampilan dalam aplikasi yang bertugas menampilkan detail dari sebuah negara yang dipilih oleh pengguna. Fragment ini menggunakan View Binding agar bisa mengakses elemen-elemen layout fragment_detail.xml dengan lebih aman dan efisien tanpa `findViewById()`.

Pertama-tama, dibuat variabel _binding yang akan menyimpan objek binding sementara fragment aktif. Nilai ini kemudian diakses melalui binding yang aman karena sudah dipastikan tidak null. Binding ini di-inflate pada fungsi `onCreateView()`, lalu digunakan untuk menampilkan isi layout fragment tersebut.

Lalu, di dalam `onViewCreated()`, fragment ini mengambil data yang dikirim dari fragment sebelumnya menggunakan arguments. Data yang diambil berupa `namaNegara` (nama negara), `alasan` (alasan memilih negara tersebut), dan `gambarResId` (ID gambar dari drawable). Kalau data tidak ditemukan, maka akan ditampilkan nilai default seperti "Tidak ada nama" atau `gambar default R.drawable.japan`.

Setelah data berhasil diambil, semuanya ditampilkan ke tampilan: gambar dimunculkan dengan setImageResource, nama negara ditaruh di `tvCountryName`, dan alasannya di `tvCountryReason`. Terakhir, pada `onDestroyView()`, binding di-null-kan agar tidak menyebabkan memory leak.

4. NegaraAdapter.kt

Kode di atas adalah isi dari kelas `NegaraAdapter`, yaitu adapter untuk `RecyclerView` yang menampilkan daftar negara. Adapter ini bertugas mengatur bagaimana data ditampilkan di setiap item daftar. NegaraAdapter menerima dua parameter: `listNegara` (daftar data negara dari model NegaraModel) dan `navController` (yang digunakan untuk navigasi ke tampilan detail). Di dalamnya ada kelas NegaraViewHolder sebagai inner class, yang menyimpan binding dari layout item_negara.xml. Binding ini digunakan untuk mengakses elemen UI dalam tiap item list tanpa harus menggunakan `findViewById()`.

Fungsi `onCreateViewHolder()` dipanggil saat `RecyclerView` ingin membuat tampilan baru. Di sini, layout item negara di-inflate dan dibungkus ke dalam NegaraViewHolder. Kemudian, fungsi `onBindViewHolder()` akan mengisi data ke dalam item berdasarkan posisi item dalam list. Misalnya, nama negara ditampilkan ke tvItemName, gambar negara ke `imgItemPhoto`.

Ketika tombol Wikipedia (btnWiki) diklik, aplikasi akan membuka link Wikipedia dari negara tersebut di browser, menggunakan Intent.`ACTION_VIEW` dan mengonversi URL-nya menjadi Uri. Lalu ada tombol buttonDetail, yang ketika diklik akan membawa pengguna ke `DetailFragment` dengan membawa data negara tersebut menggunakan Bundle. Data yang dikirim meliputi gambar, nama negara, dan alasan memilih negara tersebut. Navigasi dilakukan lewat `navController.navigate`() menuju detailFragment.

5. NegaraModel.kt

Kode di atas adalah deklarasi dari NegaraModel, yaitu sebuah data class di Kotlin yang digunakan untuk merepresentasikan data dari satu negara dalam aplikasi. Karena ini data class, maka otomatis Kotlin akan menyediakan fungsi-fungsi standar seperti `toString()`, `equals()`, dan `hashCode()` secara otomatis, yang sangat membantu untuk keperluan data.

Di dalam NegaraModel, ada lima properti yang mewakili informasi tentang negara tersebut:

- `nama`: berisi nama negara (misalnya "Jepang"),
- `tahun`: berisi informasi tahun kunjungan atau rencana (meskipun di kode sebelumnya belum digunakan),

- `alasan`: alasan kenapa negara itu dipilih,
- `gambarResId`: menyimpan ID resource dari gambar negara (biasanya dari folder res/drawable)
- `wikiUrl`: berisi link ke Wikipedia negara tersebut yang nantinya bisa dibuka lewat browser.

Model ini nantinya digunakan oleh NegaraAdapter untuk ditampilkan ke tampilan list dan juga dikirim ke `DetailFragment`. Jadi, bisa dibilang `NegaraModel` ini adalah blueprint atau cetakan dari setiap item negara yang ditampilkan di aplikasi.

6. NegaraViewModel.kt

Kode di atas adalah isi dari `NegaraViewModel`, yaitu kelas yang berfungsi sebagai jembatan antara data dan tampilan (UI) dalam arsitektur MVVM (`Model-View-ViewModel`). Kelas ini menuruni `AndroidViewModel`, yang artinya ia bisa mengakses Application langsung, berguna ketika kita butuh akses ke resource seperti string array atau drawable yang ada di res. Di dalamnya, ada fungsi `loadNegara()` yang akan memuat data dari file `strings.xml` dan mengembalikannya sebagai daftar objek NegaraModel. Data yang dimuat meliputi:

- `namaNegara`: array string yang berisi nama-nama negara,
- `tahun`: array string tahun kunjungan atau keinginan mengunjungi,
- `alasan`: array string yang menjelaskan alasan memilih negara tersebut,
- `link`: array string berisi URL Wikipedia tiap negara,
- `gambar`: daftar resource ID gambar negara dari folder drawable.

Fungsi ini kemudian menggunakan indices.map untuk menggabungkan data-data dari array tadi berdasarkan indeks yang sama, sehingga setiap elemen array (misalnya nama ke-0, tahun ke-0, dan seterusnya) dikombinasikan menjadi satu objek `NegaraModel`. Hasil akhirnya adalah sebuah list berisi data lengkap tiap negara yang siap ditampilkan di `RecyclerView` melalui `NegaraAdapter`.

7. Activity_main.xml

XML di atas merupakan bagian dari layout aplikasi Android yang ditulis dalam bahasa XML, dan biasanya digunakan untuk menentukan tampilan antarmuka pengguna (UI). Layout ini menggunakan `LinearLayout` sebagai wadah utamanya dengan orientasi vertikal, yang berarti semua elemen di dalamnya akan ditampilkan dari atas ke bawah. Pertama, ada sebuah elemen TextView yang digunakan untuk menampilkan teks `"Selamat datang di DreamsTravel!".` Teks ini ditampilkan di tengah secara horizontal (`layout_gravity="center_horizontal")` dan diberi margin sebesar `16dp` agar tidak terlalu rapat dengan elemen lain. Ukuran teksnya diatur menjadi `18sp` agar cukup besar dan mudah dibaca. Kemudian, di bawah `TextView`, terdapat `FragmentContainerView` yang berfungsi sebagai tempat untuk menampilkan fragment navigasi. `FragmentContainerView` ini menggunakan NavHostFragment dari library Jetpack Navigation, yang memungkinkan kita untuk mengatur navigasi antar fragment di dalam aplikasi. `Properti app:defaultNavHost="true"` memastikan bahwa fragment ini akan menangani aksi navigasi secara default, dan `app:navGraph="@navigation/nav_graph"` menunjukkan file grafik navigasi yang digunakan untuk menentukan alur berpindah antar fragment dalam aplikasi.

8. Fragment_detail.xml

XML di atas merupakan layout tampilan detail pada aplikasi Android yang menggunakan `ConstraintLayout` sebagai root layout-nya. `ConstraintLayout` ini memungkinkan setiap elemen UI untuk diatur posisinya secara fleksibel dan presisi dengan menghubungkannya ke elemen lain atau ke parent layout. Layout ini biasanya digunakan agar tampilan lebih responsif di berbagai ukuran layar. Pertama, terdapat sebuah `ImageView` dengan ID `imgCountry` yang berfungsi untuk menampilkan gambar, misalnya gambar bendera atau pemandangan dari suatu negara. Ukuran gambar ini diatur menjadi `200dp x 200dp` dan ditempatkan di tengah atas layar dengan margin atas `32dp`. Penempatan ini diatur menggunakan properti constraint seperti `app:layout_constraintTop_toTopOf="parent"` dan `app:layout_constraintStart_toStartOf="parent",` sehingga posisinya

selalu berada di bagian atas dan tengah secara horizontal. Di bawah gambar, ada `TextView` dengan ID `tvCountryName` yang digunakan untuk menampilkan nama negara. Teks ini akan muncul tepat di bawah gambar karena dihubungkan menggunakan `layout_constraintTop_toBottomOf="@id/imgCountry"`. Teks ini juga dibuat agar berada di tengah secara horizontal. Terakhir, ada lagi `TextView` dengan `ID tvCountryReason` yang menampilkan alasan kenapa negara tersebut menarik atau layak dikunjungi. Sama seperti elemen sebelumnya, teks ini diletakkan di bawah `tvCountryName` dan juga diatur agar berada di tengah.

9. Fragment_home.xml

XML di atas adalah layout untuk sebuah tampilan dalam aplikasi Android yang menggunakan `ConstraintLayout` sebagai layout utamanya. Di dalam layout ini hanya terdapat satu komponen, yaitu `RecyclerView`, yang merupakan elemen penting ketika kita ingin menampilkan daftar data dalam bentuk scroll, seperti daftar negara, daftar tempat wisata, atau item lainnya.

`RecyclerView` di sini memiliki `ID rvNegara`, dan ukurannya diatur agar memenuhi seluruh layar dengan menggunakan `layout_width="0dp"` dan `layout_height="0dp"` lalu dihubungkan ke semua sisi parent menggunakan constraint (Top, Bottom, Start, End). Ini artinya `RecyclerView` akan membentang dari atas sampai bawah dan dari kiri ke kanan layar.

`Properti clipToPadding="false"` berarti elemen di dalam daftar bisa terlihat meskipun berada di area padding, dan padding=`"8dp"` memberikan jarak di sekeliling isi daftar agar tidak terlalu mepet ke tepi layar. Selain itu, t`ools:listitem="@layout/item_negara"` adalah petunjuk khusus untuk Android Studio agar saat kita melihat preview-nya, akan ditampilkan contoh isi daftar yang berasal dari layout item bernama item_negara.

10. Item_negara.xml

XML di atas adalah layout untuk sebuah item dalam daftar (biasanya digunakan dalam `RecyclerView`) dan dirancang menggunakan komponen `CardView` untuk memberikan tampilan yang elegan dan modern, seperti kartu dengan sudut melengkung dan bayangan

(elevation). `CardView` ini berisi `ConstraintLayout` yang digunakan untuk mengatur posisi elemen-elemen di dalamnya agar tetap responsif di berbagai ukuran layar.

Di dalam `CardView`, terdapat tiga elemen utama. Pertama, ada `ShapeableImageView` yang digunakan untuk menampilkan gambar dari item, misalnya gambar negara. Gambar ini diposisikan di sebelah kiri, dan menggunakan `centerCrop` agar gambar terlihat rapi dan memenuhi ruang yang disediakan tanpa merusak proporsi.

Kedua, terdapat `TextView` yang akan menampilkan nama negara (atau item lainnya). Teks ini diletakkan di samping kanan gambar dan diatur agar tampil tegas (dengan ukuran `18sp` dan huruf tebal). Penempatan teks ini sangat fleksibel karena menggunakan `ConstraintLayout` untuk mengatur jarak dengan gambar di sebelah kiri dan batas layout di kanan. Ketiga, terdapat dua buah Button. Yang pertama adalah tombol "Detail" (`button_detail`) yang biasanya digunakan untuk melihat informasi lebih lengkap tentang item tersebut. Di sebelah kanannya, ada tombol "Wiki" (btn_wiki) yang kemungkinan besar mengarahkan pengguna ke halaman Wikipedia terkait item tersebut. Kedua tombol ini ditempatkan sejajar di bawah `TextView` dan memiliki padding agar nyaman ditekan.

11. Nav_graph.xml

XML di atas adalah bagian dari navigation graph dalam aplikasi Android yang menggunakan Jetpack Navigation Component. Navigation graph ini digunakan untuk mengatur dan mengelola alur perpindahan antar halaman atau fragment di dalam aplikasi. Jadi, kita bisa menentukan dari fragment mana ke fragment mana pengguna bisa berpindah, dan bagaimana arah alurnya. Dalam file ini, ditentukan bahwa homeFragment adalah halaman awal yang akan ditampilkan ketika aplikasi dibuka, ditandai dengan `app:startDestination="@id/homeFragment".` `homeFragment` merujuk pada `com.example.dreamstravel.HomeFragment`, yang tampil menggunakan layout `fragment_home.` Di dalamnya juga ada sebuah action, yaitu `action_homeFragment_to_detailFragment`, yang berarti dari `homeFragment` pengguna bisa melakukan navigasi ke `detailFragment`. Lalu, ada `detailFragment` yang merupakan tujuan navigasi, dengan nama `com.example.dreamstravel.DetailFragment`, dan menggunakan layout

`fragment_detail`. Fragment ini akan ditampilkan saat aksi navigasi dilakukan dari halaman utama.

12. Strings.xml

XML di atas adalah file strings.xml yang digunakan dalam proyek Android untuk menyimpan berbagai teks atau data dalam bentuk string, agar tidak ditulis langsung (hardcoded) di layout atau kode program. Hal ini sangat membantu dalam pengelolaan aplikasi, terutama untuk mendukung banyak bahasa (lokalisasi) atau saat ingin mengubah isi teks tanpa menyentuh file layout atau Java/Kotlin-nya. Pertama, terdapat beberapa string biasa, seperti app_name untuk nama aplikasi yaitu `"DreamsTravel",` serta `btn_detail`, `btn_wiki`, dan `img_desc` yang masing-masing berisi teks untuk tombol dan deskripsi gambar. Kemudian, ada juga beberapa string-array, yaitu array yang berisi kumpulan data teks. Misalnya, list_negara berisi nama-nama negara impian seperti Jepang, Swiss, hingga Prancis. Ini biasanya ditampilkan dalam bentuk daftar atau pilihan di aplikasi.

Ada juga array `list_tahun` yang memuat beberapa tahun dari 2025 sampai 2029, kemungkinan digunakan untuk memilih tahun impian untuk berangkat. Selanjutnya, `data_gambar` adalah array yang berisi referensi gambar dari folder drawable, digunakan untuk menampilkan gambar negara sesuai nama-nama di atas. Selain itu, ada list_alasan yang menjelaskan alasan kenapa pengguna ingin mengunjungi masing-masing negara—isnya seperti motivasi pribadi, misalnya karena budaya, musik, agama, atau ingin bertemu selebriti. Terakhir, ada `list_link` yang berisi tautan Wikipedia dari masing-masing negara, digunakan saat pengguna ingin tahu lebih detail dengan membuka informasi lewat web.

SOAL 2

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

A. Pembahasan

Alasan RecyclerView masih banyak digunakan walaupun kodenya lebih panjang dan terkesan boiler-plate adalah karena:

1. RecyclerView adalah bagian dari View system (lama) yang sudah stabil dan banyak digunakan di dunia industri. Banyak aplikasi besar yang dibangun sebelum Jetpack Compose muncul, dan mereka masih bergantung pada sistem View, termasuk RecyclerView.
2. Kompatibilitas dan dokumentasi luas. RecyclerView sudah ada sejak lama, jadi dokumentasinya lengkap, komunitasnya besar, dan banyak library serta tools yang mendukungnya. Kalau kamu butuh referensi, StackOverflow dan GitHub penuh dengan contoh.
3. RecyclerView sangat fleksibel dan bisa dikustomisasi. Meski kelihatannya ribet, RecyclerView bisa digunakan untuk berbagai jenis tampilan: daftar sederhana, grid, carousel, dan bahkan chat layout, hanya dengan mengubah layout manager atau adapter-nya.
4. Tidak semua proyek menggunakan Jetpack Compose. LazyColumn adalah bagian dari Compose, yang merupakan pendekatan UI yang relatif baru. Banyak perusahaan masih memakai sistem View karena belum semua tim siap untuk migrasi besar-besaran ke Compose.
5. Compose belum sepenuhnya matang untuk semua kasus. Meskipun Compose (dan LazyColumn) lebih singkat dan modern, dalam beberapa kasus performa atau dukungan komponennya masih kalah dengan View system, tergantung kompleksitas aplikasi.

B. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/annacoded/Pemrograman-Mobile

MODUL 4 : VIEW MODEL AND DEBUGGING

SOAL 1

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

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

A. Source Code

1. MainActivity.kt

Tabel 16. Source Code Jawaban Soal 1 Modul 4 MainActivity.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import androidx.activity.compose.setContent
5  import androidx.activity.enableEdgeToEdge
6  import androidx.compose.foundation.layout.fillMaxSize
7  import androidx.compose.foundation.layout.padding
8  import androidx.compose.material3.Scaffold
```

```
9  import androidx.compose.material3.Text
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.tooling.preview.Preview
13 import androidx.appcompat.app.AppCompatActivity
14 import androidx.navigation.fragment.NavHostFragment
15 import com.example.dreamstravel.databinding.ActivityMainBinding
16 import com.example.dreamstravel.ui.theme.DreamsTravelTheme
17
18 class MainActivity : AppCompatActivity() {
19
20     private lateinit var binding: ActivityMainBinding   // ←
   Binding object
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         val                  navHostFragment                  =
   supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
   as NavHostFragment
29         val navController = navHostFragment.navController
30     }
31 }
```

2. DetailFragment.kt

Tabel 17. Source Code Jawaban Soal 1 Modul 4 DetailFragment.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import androidx.activity.compose.setContent
5  import androidx.activity.enableEdgeToEdge
6  import androidx.compose.foundation.layout.fillMaxSize
7  import androidx.compose.foundation.layout.padding
8  import androidx.compose.material3.Scaffold
9  import androidx.compose.material3.Text
10 import androidx.compose.runtime.Composable
11 import androidx.compose.ui.Modifier
12 import androidx.compose.ui.tooling.preview.Preview
13 import androidx.appcompat.app.AppCompatActivity
14 import androidx.navigation.fragment.NavHostFragment
15 import com.example.dreamstravel.databinding.ActivityMainBinding
16 import com.example.dreamstravel.ui.theme.DreamsTravelTheme
```

```
17
18 class MainActivity : AppCompatActivity() {
19
20     private lateinit var binding: ActivityMainBinding   // ←
   Binding object
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24
25         binding = ActivityMainBinding.inflate(layoutInflater)
26         setContentView(binding.root)
27
28         val                    navHostFragment                    =
   supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
   as NavHostFragment
29         val navController = navHostFragment.navController
30     }
31 }
```

3. HomeFragment.kt

Tabel 18. Source Code Jawaban Soal 1 Modul 4 HomeFragment.kt

```
1  package com.example.dreamstravel
2
3  import NegaraViewModel
4  import NegaraViewModelFactory
5  import android.content.Intent
6  import android.os.Bundle
7  import android.util.Log
8  import android.view.LayoutInflater
9  import android.view.View
10 import android.view.ViewGroup
11 import androidx.fragment.app.Fragment
12 import androidx.lifecycle.ViewModelProvider
13 import androidx.lifecycle.lifecycleScope
14 import androidx.recyclerview.widget.LinearLayoutManager
15 import
   com.example.dreamstravel.databinding.FragmentHomeBinding
16 import kotlinx.coroutines.launch
17 import androidx.core.net.toUri
18 import androidx.navigation.fragment.findNavController
19
20 class HomeFragment : Fragment() {
21     private var _binding: FragmentHomeBinding? = null
22     private val binding get() = _binding!!
23
```

```
24      private lateinit var viewModel: NegaraViewModel
25      private lateinit var adapter: NegaraAdapter
26
27      override fun onCreateView(inflater: LayoutInflater,
    container: ViewGroup?, savedInstanceState: Bundle?): View {
28          _binding = FragmentHomeBinding.inflate(inflater,
    container, false)
29          return binding.root
30      }
31
32      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
33          // Gunakan ViewModelFactory
34          val factory =
    NegaraViewModelFactory(requireActivity().application)
35          viewModel = ViewModelProvider(this,
    factory)[NegaraViewModel::class.java]
36
37          // Setup Adapter tanpa navController, dengan dua
    callback
38          adapter = NegaraAdapter(
39              listNegara = emptyList(),
40              onDetailClick = { negara ->
41                  Log.d("HomeFragment", "Tombol Detail
    ditekan: ${negara.nama}")
42                  viewModel.onItemClicked(negara)
43
44                  val bundle = Bundle().apply {
45                      putInt("gambarResId",
    negara.gambarResId)
46                      putString("namaNegara", negara.nama)
47                      putString("alasan", negara.alasan)
48                  }
49
50                  Log.d("HomeFragment", "Navigasi ke detail
    dengan: ${negara.nama}")
51
    findNavController().navigate(R.id.detailFragment, bundle)
52              },
53              onWikiClick = { negara ->
54                  Log.d("HomeFragment", "Tombol Wiki ditekan:
    ${negara.nama}")
55                  val intent = Intent(Intent.ACTION_VIEW,
    negara.wikiUrl.toUri())
56                  startActivity(intent)
57              }
58          )
```

```
59
60          binding.rvNegara.layoutManager                          =
61  LinearLayoutManager(requireContext())
            binding.rvNegara.adapter = adapter
62
63          // Observe StateFlow dari ViewModel
64          lifecycleScope.launch {
65              viewModel.listNegara.collect { negaraList ->
66                  Log.d("HomeFragment",   "Data   listNegara
67  masuk (${negaraList.size} item)")
                    adapter.updateData(negaraList)
68              }
69          }
70
71          // Panggil loadNegara agar data dimuat
72          viewModel.loadNegara()
73      }
74
75      override fun onDestroyView() {
76          super.onDestroyView()
77          _binding = null
78      }
79  }
80
```

4. NegaraAdapter.kt

Tabel 19. Source Code Jawaban Soal 1 Modul 4 NegaraAdapter.kt

```
1   package com.example.dreamstravel
2
3   import android.content.Intent
4   import androidx.core.net.toUri
5   import android.view.LayoutInflater
6   import android.view.ViewGroup
7   import androidx.recyclerview.widget.RecyclerView
8   import com.example.dreamstravel.databinding.ItemNegaraBinding
9
10  class NegaraAdapter(
11      private var listNegara: List<NegaraModel>,
12      private val onDetailClick: (NegaraModel) -> Unit,
13      private val onWikiClick: (NegaraModel) -> Unit
14  ) : RecyclerView.Adapter<NegaraAdapter.NegaraViewHolder>() {
15
16      inner class NegaraViewHolder(val binding: ItemNegaraBinding)
    : RecyclerView.ViewHolder(binding.root)
```

```
17      override fun onCreateViewHolder(parent: ViewGroup, viewType:
18  Int): NegaraViewHolder {
            val                          binding                          =
19  ItemNegaraBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
            return NegaraViewHolder(binding)
20      }
21
22      override   fun   onBindViewHolder(holder:   NegaraViewHolder,
23  position: Int) {
            val negara = listNegara[position]
24          holder.binding.tvItemName.text = negara.nama
25
26  holder.binding.imgItemPhoto.setImageResource(negara.gambarResId)

27          holder.binding.btnWiki.setOnClickListener {
28              onWikiClick(negara)
29          }
30
31          holder.binding.buttonDetail.setOnClickListener {
32              onDetailClick(negara)
33          }
34      }
35
36      override fun getItemCount() = listNegara.size
37
38      fun updateData(newList: List<NegaraModel>) {
39          listNegara = newList
40          notifyDataSetChanged()
41      }
42  }
43
```

5. NegaraModel.kt

Tabel 20. Source Code Jawaban Soal 1 Modul 4 NegaraModel.kt

```
1   package com.example.dreamstravel
2
3   data class NegaraModel(
4       val nama: String,
5       val tahun: String,
6       val alasan: String,
7       val gambarResId: Int,
8       val wikiUrl: String
9   )
```

6. NegaraViewModel.kt

Tabel 21. Source Code Jawaban Soal 1 Modul 4 NegaraViewModel.kt

```
1  import android.app.Application
2  import android.util.Log
3  import androidx.lifecycle.AndroidViewModel
4  import androidx.lifecycle.viewModelScope
5  import kotlinx.coroutines.flow.MutableStateFlow
6  import kotlinx.coroutines.flow.StateFlow
7  import kotlinx.coroutines.launch
8  import com.example.dreamstravel.NegaraModel
9  import com.example.dreamstravel.R
10
11 class NegaraViewModel(application: Application) :
   AndroidViewModel(application) {
12
13     private val context =
   getApplication<Application>().applicationContext
14
15     private val _listNegara =
   MutableStateFlow<List<NegaraModel>>(emptyList())
16     val listNegara: StateFlow<List<NegaraModel>> get() =
   _listNegara
17
18     private val _selectedNegara =
   MutableStateFlow<NegaraModel?>(null)
19     val selectedNegara: StateFlow<NegaraModel?> get() =
   _selectedNegara
20
21     fun loadNegara() {
22         viewModelScope.launch {
23             val namaNegara =
   context.resources.getStringArray(R.array.list_negara)
24             val tahun =
   context.resources.getStringArray(R.array.list_tahun)
25             val alasan =
   context.resources.getStringArray(R.array.list_alasan)
26             val link =
   context.resources.getStringArray(R.array.list_link)
27             val gambar = listOf(
28                 R.drawable.japan,
29                 R.drawable.swiss,
30                 R.drawable.korsel,
31                 R.drawable.arabsaudi,
32                 R.drawable.france
33             )
34
```

```
35              val negaraList = namaNegara.indices.map { i ->
36                  NegaraModel(
37                      namaNegara[i],
38                      tahun[i],
39                      alasan[i],
40                      gambar[i],
41                      link[i]
42                  )
43              }
44
45              _listNegara.value = negaraList
46              Log.d("NegaraViewModel", "List negara dimuat:
    ${negaraList.size} item")
47          }
48      }
49
50      fun onItemClicked(negara: NegaraModel) {
51          _selectedNegara.value = negara
52          Log.d("NegaraViewModel",      "Negara      dipilih:
    ${negara.nama}")
53      }
54  }
```

7. ViewModelFactory.kt

Tabel 22. Source Code Jawaban Soal 1 Modul 4 ViewModelFactory

```
1   import android.app.Application
2   import androidx.lifecycle.ViewModel
3   import androidx.lifecycle.ViewModelProvider
4
5   class   NegaraViewModelFactory(private   val   application:
    Application) : ViewModelProvider.Factory {
6       override   fun   <T   :   ViewModel>   create(modelClass:
    Class<T>): T {
7           if
8   (modelClass.isAssignableFrom(NegaraViewModel::class.java))
    {
9               return NegaraViewModel(application) as T
10          }
11          throw  IllegalArgumentException("Unknown  ViewModel
    class")
12      }
13  }
```

8. Activity_main.xml

Tabel 23. Source Code Jawaban Soal 1 Modul 4 Activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical">
8
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Selamat datang di DreamsTravel!"
13         android:textSize="18sp"
14         android:layout_gravity="center_horizontal"
15         android:layout_margin="16dp"/>
16
17     <androidx.fragment.app.FragmentContainerView
18         android:id="@+id/nav_host_fragment"
19
20 android:name="androidx.navigation.fragment.NavHostFragment"
           android:layout_width="match_parent"
21         android:layout_height="match_parent"
22         app:defaultNavHost="true"
23         app:navGraph="@navigation/nav_graph" />
24 </LinearLayout>
```

9. Fragment_detail.xml

Tabel 24. Source Code Jawaban Soal 1 Modul 4 Fragment_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".DetailFragment">
8
9      <ImageView
10         android:id="@+id/imgCountry"
11         android:layout_width="200dp"
12         android:layout_height="200dp"
```

```
13          android:layout_marginTop="32dp"
14          android:layout_centerHorizontal="true"
15          app:layout_constraintTop_toTopOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent" />
18
19      <TextView
20          android:id="@+id/tvCountryName"
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:textSize="40sp"

24  app:layout_constraintTop_toBottomOf="@id/imgCountry"
25          app:layout_constraintStart_toStartOf="parent"
26          app:layout_constraintEnd_toEndOf="parent"
27          android:text="Country Name"/>
28
29      <TextView
30          android:id="@+id/tvCountryReason"
31          android:layout_width="match_parent"
32          android:layout_height="wrap_content"
33          android:justificationMode="inter_word"
34          android:textAlignment="viewStart"
35          android:layout_margin="16dp"
36          android:textSize="16sp"
37          android:layout_marginTop="16dp"
38
    app:layout_constraintTop_toBottomOf="@id/tvCountryName"
39          app:layout_constraintStart_toStartOf="parent"
40          app:layout_constraintEnd_toEndOf="parent"
41          android:text="Reason"/>
42  </androidx.constraintlayout.widget.ConstraintLayout>
```

10. Fragment_home.xml

Tabel 25. Source Code Jawaban Soal 1 Modul 4 Fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
    <androidx.constraintlayout.widget.ConstraintLayout
2   xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
3       android:layout_width="match_parent"
4       android:layout_height="match_parent"
5       xmlns:tools="http://schemas.android.com/tools">
6
7       <androidx.recyclerview.widget.RecyclerView
8           android:id="@+id/rvNegara"
```

```
9           android:layout_width="0dp"
10          android:layout_height="0dp"
11          android:clipToPadding="false"
12          android:padding="8dp"
13          android:contentDescription="cute"
14          app:layout_constraintTop_toTopOf="parent"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintStart_toStartOf="parent"
17          app:layout_constraintEnd_toEndOf="parent"
18          tools:listitem="@layout/item_negara" />
19  </androidx.constraintlayout.widget.ConstraintLayout>
20
```

11. Item_negara.xml

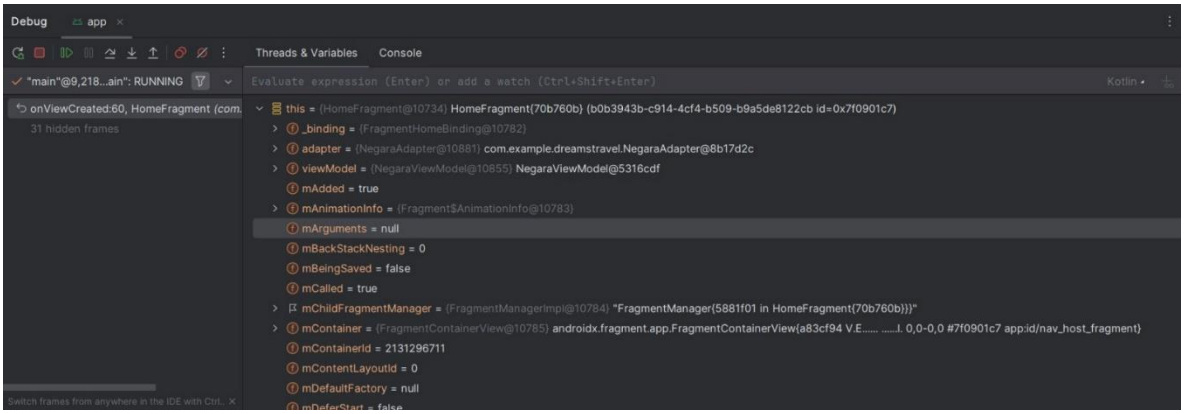
Tabel 26. Source Code Jawaban Soal 1 Modul 4 Item_negara.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:card_view="http://schemas.android.com/apk/res-
    auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       android:layout_width="match_parent"
7       android:layout_height="wrap_content"
8       android:layout_margin="@dimen/card_margin"
9       app:cardCornerRadius="@dimen/card_radius"
10      app:cardElevation="4dp">
11
12      <androidx.constraintlayout.widget.ConstraintLayout
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content"
15          android:padding="@dimen/card_padding">
16
17
    <com.google.android.material.imageview.ShapeableImageView
18              android:id="@+id/img_item_photo"
19              android:layout_width="80dp"
20              android:layout_height="100dp"
21              android:scaleType="centerCrop"
22              app:layout_constraintStart_toStartOf="parent"
23              app:layout_constraintTop_toTopOf="parent"
24
    app:layout_constraintBottom_toBottomOf="parent"
25              android:contentDescription="@string/img_desc"
    />
```

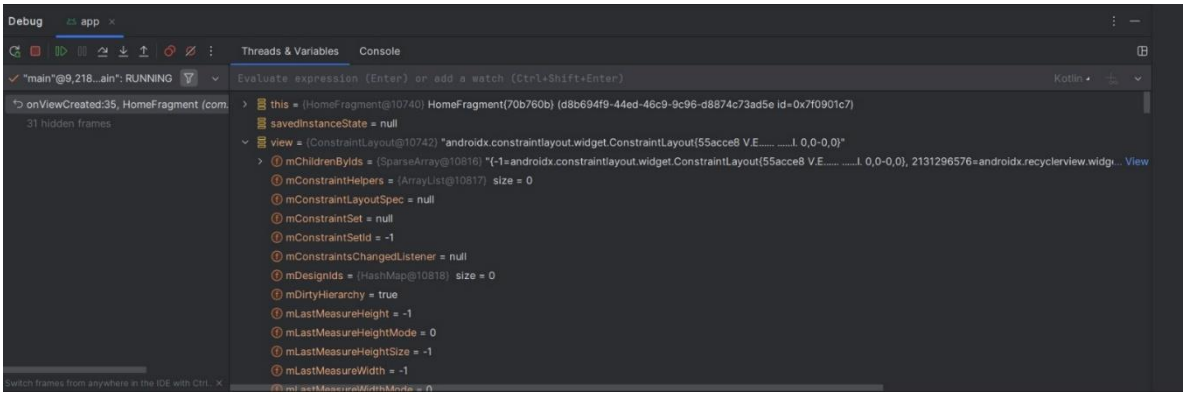
```
26
27          <TextView
28              android:id="@+id/tv_item_name"
29              android:layout_width="0dp"
30              android:layout_height="wrap_content"
31              android:textSize="18sp"
32              android:textStyle="bold"
33              android:textColor="@android:color/black"
34
    app:layout_constraintStart_toEndOf="@id/img_item_photo"
35              app:layout_constraintTop_toTopOf="parent"
36              app:layout_constraintEnd_toEndOf="parent"
37              app:layout_constraintHorizontal_bias="0"
38              android:layout_marginStart="8dp"
39              android:layout_marginEnd="8dp"
40              tools:text="Jepang" />
41
42          <Button
43              android:id="@+id/button_detail"
44              android:layout_width="wrap_content"
45              android:layout_height="wrap_content"
46              android:text="@string/btn_detail"
47              android:minHeight="48dp"
48              android:minWidth="48dp"
49
    android:paddingHorizontal="@dimen/button_padding"
50
    app:layout_constraintTop_toBottomOf="@id/tv_item_name"
51
    app:layout_constraintStart_toStartOf="@id/tv_item_name"
52
    app:layout_constraintBottom_toBottomOf="parent" />
53
54          <Button
55              android:id="@+id/btn_wiki"
56              android:layout_width="wrap_content"
57              android:layout_height="wrap_content"
58              android:text="@string/btn_wiki"
59              android:minHeight="48dp"
60              android:minWidth="48dp"
61
    android:paddingHorizontal="@dimen/button_padding"
62
    app:layout_constraintTop_toBottomOf="@id/tv_item_name"
63
    app:layout_constraintStart_toEndOf="@id/button_detail"
64
```

```
65  app:layout_constraintBottom_toBottomOf="parent"
66              android:layout_marginStart="8dp" />
67
68      </androidx.constraintlayout.widget.ConstraintLayout>
    </androidx.cardview.widget.CardView>
```

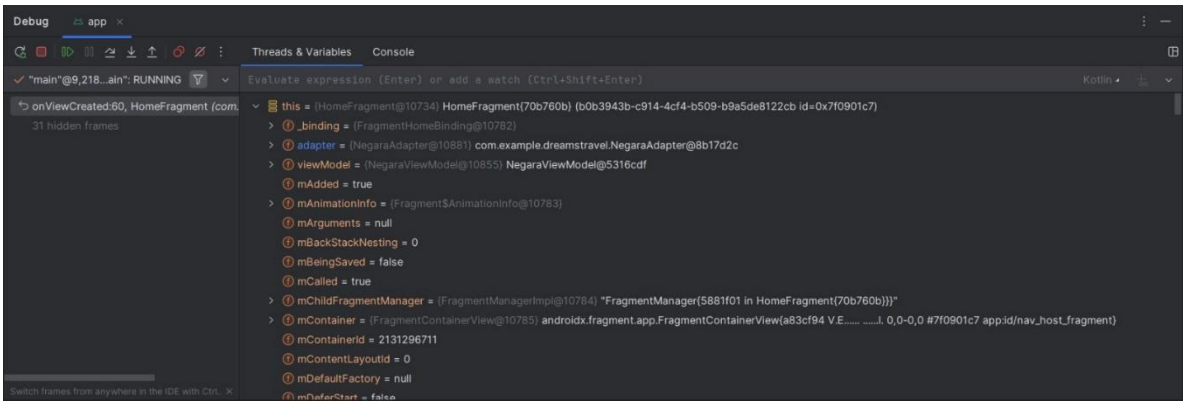
B. Output Program



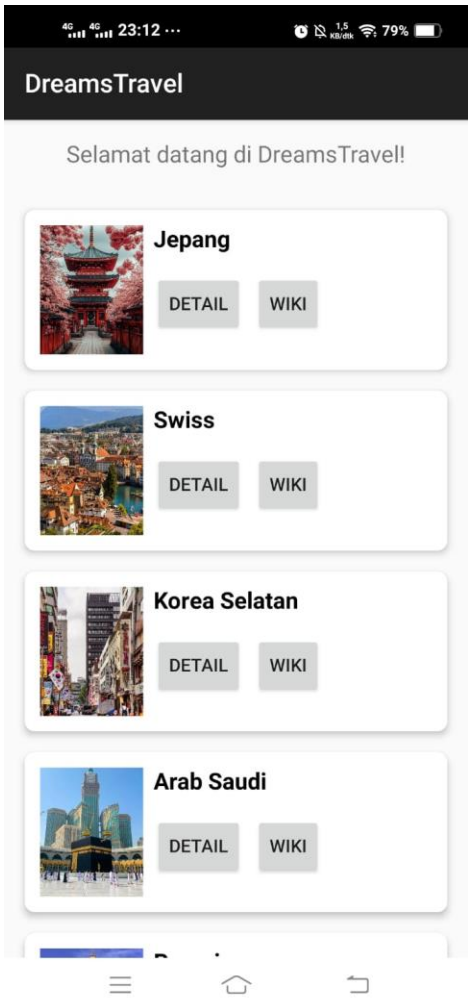
Gambar 10. Debugger Step Into



Gambar 11. Debugger Step Over



Gambar 12. Debugger Step Over



Gambar 13. Tampilan Home

Gambar 14. Tampilan Detail

C. Pembahasan

1. MainActivity.kt:

Kode di atas adalah bagian dari aplikasi Android yang ditulis menggunakan Kotlin dan Jetpack Compose. Kelas `MainActivity` merupakan aktivitas utama yang dijalankan saat aplikasi dibuka. Di dalam `onCreate()`, `ActivityMainBinding` digunakan untuk menghubungkan file XML `activity_main.xml` ke kode Kotlin, sehingga elemen UI bisa diakses langsung. Kemudian, sistem navigasi menggunakan NavHostFragment diinisialisasi agar aplikasi bisa berpindah-pindah antar fragment.

2. DetailFragment.kt

Kode di atas adalah kelas `DetailFragment`, yaitu salah satu halaman (fragment) dalam aplikasi Android yang menampilkan detail suatu negara. Fragment ini menggunakan view binding (`FragmentDetailBinding`) untuk menghubungkan elemen-elemen di layout XML dengan kode Kotlin. Saat fragment dibuat (`onCreateView`), layout-nya di-inflate dan disimpan ke dalam variabel binding. Di `onViewCreated`, fragment mengambil data seperti nama negara, alasan, dan gambar dari argument (data yang dikirim dari fragment sebelumnya). Data ini kemudian ditampilkan ke tampilan UI. Terakhir, di `onDestroyView`, binding di-set ke null untuk menghindari memory leak.

3. **HomeFragment.kt**

   Kode di atas merupakan implementasi dari `HomeFragment,` yaitu halaman utama yang menampilkan daftar negara yang ingin dikunjungi. Fragment ini menggunakan `FragmentHomeBinding` untuk menghubungkan layout XML ke kode Kotlin, serta menggunakan `ViewModel` untuk mengelola data negara. Di dalam `onViewCreated`, `NegaraViewModel` diinisialisasi menggunakan `ViewModelFactory`, lalu `NegaraAdapter` digunakan untuk menampilkan data dalam bentuk daftar (RecyclerView). Adapter ini punya dua aksi: ketika tombol Detail ditekan, aplikasi akan menavigasi ke `DetailFragment` sambil mengirim data negara melalui `Bundle`; sedangkan jika tombol Wiki ditekan, akan membuka halaman Wikipedia negara tersebut di browser. Data negara dimuat dan diamati menggunakan `StateFlow`, dan jika ada perubahan, RecyclerView akan diperbarui. Terakhir, binding dibersihkan di `onDestroyView` untuk menghindari memory leak.

4. **NegaraAdapter.kt**

   Kode di atas adalah kelas `NegaraAdapter` yang digunakan untuk menampilkan daftar negara dalam bentuk RecyclerView. Adapter ini menerima data berupa list dari `NegaraModel`, serta dua aksi saat tombol ditekan: `onDetailClick` dan `onWikiClick`. Di dalamnya terdapat `ViewHolder` yang menghubungkan layout `item_negara.xml` dengan data negara. Di `onBindViewHolder`, setiap item akan diisi dengan nama dan gambar negara, serta dua tombol: satu untuk membuka

link Wikipedia (`btnWiki`), dan satu lagi untuk melihat detail negara (`buttonDetail`). Fungsi `updateData` digunakan untuk memperbarui daftar negara saat datanya berubah.

5. **NegaraModel.kt**

   Kode di atas adalah data class bernama `NegaraModel` yang berfungsi untuk menyimpan data dari sebuah negara. Setiap objek `NegaraModel` berisi informasi seperti nama negara (`nama`), tahun impian untuk dikunjungi (`tahun`), alasan ingin mengunjungi negara tersebut (`alasan`), ID gambar dari sumber daya aplikasi (`gambarResId`), dan link Wikipedia negara tersebut (`wikiUrl`). Kelas ini biasanya digunakan sebagai model data yang akan ditampilkan di tampilan aplikasi, seperti dalam RecyclerView.

6. **NegaraViewModel.kt**

   Kode di atas adalah kelas `NegaraViewModel`, yang berfungsi sebagai penghubung antara data dan tampilan (UI) dalam arsitektur MVVM (Model-View-ViewModel). Kelas ini mewarisi dari `AndroidViewModel` karena membutuhkan akses ke konteks aplikasi. Di dalamnya ada dua `StateFlow`: satu untuk menyimpan daftar negara (`listNegara`), dan satu lagi untuk menyimpan data negara yang dipilih (`selectedNegara`). Fungsi `loadNegara`() dipanggil untuk memuat data dari resource (seperti array nama, tahun, alasan, gambar, dan link) lalu menggabungkannya menjadi list `NegaraModel`. List ini kemudian diberikan ke UI melalui StateFlow. Ada juga fungsi `onItemClicked()` yang menyimpan negara yang diklik pengguna.

7. **ViewModelFactory.kt**

   Kode di atas adalah kelas `NegaraViewModel` yang digunakan untuk mengelola data negara dalam aplikasi Android dengan arsitektur MVVM. Kelas ini mewarisi `AndroidViewModel` agar bisa mengakses resource aplikasi seperti array dari `strings.xml`. Di dalamnya terdapat dua `StateFlow`: yang pertama `_listNegara` menyimpan daftar semua negara, dan yang kedua `_selectedNegara` menyimpan negara yang dipilih. Fungsi `loadNegara()` digunakan untuk mengambil data nama, tahun, alasan, link Wikipedia, dan gambar

dari resource, lalu digabungkan menjadi objek `NegaraModel` dan dimasukkan ke StateFlow. Fungsi `onItemClicked()` dipakai untuk menyimpan data negara yang dipilih pengguna.

8. **Activity_main.xml**

   Kode XML di atas merupakan layout utama dari aplikasi yang menggunakan `LinearLayout` sebagai wadah vertikal. Di dalamnya terdapat `TextView` yang menampilkan sambutan "Selamat datang di DreamsTravel!" di tengah atas layar dengan ukuran teks 18sp dan margin 16dp. Di bawahnya, ada `FragmentContainerView` yang berfungsi sebagai tempat untuk menampilkan fragment-fragment lain berdasarkan navigasi. Komponen ini menggunakan `NavHostFragment` dari Jetpack Navigation dan diarahkan ke file `nav_graph.xml` yang berisi alur navigasi antar fragment

9. **Fragment_detail.xml**

   Kode XML di atas adalah layout untuk tampilan detail negara pada `DetailFragment`. Layout ini menggunakan `ConstraintLayout`, yang memungkinkan pengaturan posisi elemen UI secara fleksibel. Di dalamnya ada tiga komponen utama: pertama, `ImageView` dengan ID `imgCountry` yang akan menampilkan gambar negara; kedua, `TextView` dengan ID `tvCountryName` yang menampilkan nama negara dengan ukuran teks besar (40sp); dan ketiga, `TextView` dengan ID `tvCountryReason` yang menampilkan alasan pengguna ingin mengunjungi negara tersebut, dengan teks rata kiri dan margin agar terlihat rapi. Semua elemen diposisikan menggunakan constraint, agar tampilan tetap responsif di berbagai ukuran layar.

10. **Fragment_home.xml**

    Kode XML di atas adalah layout untuk `HomeFragment`, yang menggunakan `ConstraintLayout` sebagai wadah utama. Di dalamnya terdapat sebuah `RecyclerView` dengan ID `rvNegara`, yang digunakan untuk menampilkan daftar negara secara scrollable. Ukurannya dibuat memenuhi layar (`0dp` untuk width dan height) dan diatur posisinya agar menempel ke semua sisi parent menggunakan constraint. Properti `clipToPadding` diatur `false` agar isi tidak terpotong oleh

padding, dan `tools:listitem` digunakan saat preview di Android Studio untuk menunjukkan bahwa tiap item di dalam RecyclerView akan menggunakan layout `item_negara.xml`. Singkatnya, layout ini menampilkan daftar negara dengan desain responsif dan teratur

11. Item_negara.xml

Kode XML ini adalah layout untuk satu item dalam daftar negara (yang ditampilkan di `RecyclerView`). Layout ini menggunakan `CardView` supaya tampilannya seperti kartu dengan sudut melengkung dan bayangan (elevation). Di dalam `CardView` terdapat `ConstraintLayout` yang mengatur posisi elemen-elemen seperti gambar, nama negara, dan dua tombol.

Gambar negara ditampilkan lewat `ShapeableImageView`, lalu ada `TextView` untuk menampilkan nama negara. Di bawah nama negara ada dua `Button`: satu untuk membuka detail negara (`button_detail`), dan satu lagi untuk membuka link Wikipedia (`btn_wiki`). Semua elemen diposisikan dengan constraint agar tetap rapi di berbagai ukuran layar.

Penjelasan Debugger, fitur Step Into, Steo Over, dan Step Out

1. Definisi Debugger

Debugger adalah alat bantu dalam pemrograman yang digunakan untuk mencari dan memperbaiki bug (kesalahan) di dalam kode. Dengan debugger, kita bisa menjalankan program langkah demi langkah, melihat isi variabel, dan mengetahui bagian mana dari kode yang menyebabkan error.

2. Cara Menggunakan Debugger

- Set Breakpoint

→ Klik di samping baris kode (di IDE seperti Android Studio atau IntelliJ) untuk menandai titik berhenti (breakpoint).

→ Program akan berhenti sementara di baris itu saat dijalankan dalam mode debug.

- Jalankan Program dalam Mode Debug

→ Klik tombol debug (ikon bug atau debug play button).

- Gunakan fitur kontrol untuk melacak eksekusi program.

3. Fitur

1. Step Into (F7)

Masuk ke dalam fungsi yang sedang dipanggil.

Contoh: Jika ada myFunction(), debugger akan masuk ke dalam isi dari myFunction().

2. Step Over (F8)

Melewati fungsi tanpa masuk ke dalamnya.

Contoh: myFunction() tetap dijalankan, tapi debugger tidak masuk ke dalamnya — langsung ke baris berikutnya.

3. Step Out (Shift+F8)

Keluar dari fungsi saat ini dan kembali ke pemanggilnya.

Misal kamu sudah di dalam myFunction(), maka ini akan melompat keluar ke baris setelah myFunction() dipanggil.

SOAL 2

Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

1. Definisi Application class

Dalam arsitektur aplikasi Android, kelas Application adalah salah satu komponen utama yang digunakan untuk menyimpan dan mengelola status global aplikasi. Ini adalah titik awal kehidupan aplikasi Android, dan biasanya digunakan untuk inisialisasi yang harus dilakukan sekali seumur hidup aplikasi.

Application adalah kelas dasar dari sistem Android yang berisi semua komponen aplikasi (seperti Activity, Service, BroadcastReceiver, dan ContentProvider). Android membuat instance dari kelas ini sebelum membuat komponen lainnya saat aplikasi dijalankan.

2. Fungsi utama dari Application class

1. Inisialisasi Global

- Menjalankan kode yang harus diinisialisasi hanya sekali, seperti pustaka pihak ketiga (misalnya Retrofit, Dagger, Firebase).
- Contoh: konfigurasi logging, analytics, dependency injection.

2. Penyimpanan Data Global

- Menyimpan informasi atau state yang dibutuhkan di seluruh bagian aplikasi.
- Misalnya: token autentikasi, data pengguna sementara.

3. Mengakses Konteks Aplikasi

- Memberikan akses ke konteks aplikasi (ApplicationContext), yang bisa digunakan oleh berbagai komponen.

4. Lifecycle-aware Application

- Bisa digunakan bersama ProcessLifecycleOwner untuk mengetahui lifecycle aplikasi secara global.

3. Cara menggunakan Application class

1. Buat kelas yang extends Application

```
class MyApp : Application() {
    override fun onCreate() {
        super.onCreate()
```

```
            // Inisialisasi global
            Log.d("MyApp", "Application started")
        }
}
```

2. Deklarasikan di AndroidManifest.xml:

```
    <application
        android:name=".MyApp"
        ... >
        ...
</application>
```

B. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/annacoded/Pemrograman-Mobile

MODUL 5 :CONNECT TO THE INTERNET

SOAL 1

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

a Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.

b Gunakan KotlinX Serialization sebagai library JSON.

c Gunakan library seperti Coil atau Glide untuk image loading.

d API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started

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

f Gunakan caching strategy pada Room..

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

A. Source Code

1. MainActivity.kt

Tabel 27. Source Code Jawaban Soal 1 Modul 5 MainActivity.kt

```
1  package com.example.dreamstravel
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import androidx.navigation.fragment.NavHostFragment
6  import androidx.navigation.ui.setupWithNavController
7  import com.example.dreamstravel.databinding.ActivityMainBinding
8
9  class MainActivity : AppCompatActivity() {
10
11     private lateinit var binding: ActivityMainBinding
```

```
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          binding = ActivityMainBinding.inflate(layoutInflater)
16          setContentView(binding.root)
17
18          val                    navHostFragment                    =
    supportFragmentManager.findFragmentById(R.id.nav_host_fragment)
    as NavHostFragment
19          val navController = navHostFragment.navController
20
21  binding.bottomNavigation.setupWithNavController(navController)
22      }
23  }
```

2. DetailFragment.kt

Tabel 28. Source Code Jawaban Soal 1 Modul 5 DetailFragment.kt

```
 1  package com.example.dreamstravel
 2
 3  import android.os.Bundle
 4  import android.view.LayoutInflater
 5  import android.view.View
 6  import android.view.ViewGroup
 7  import androidx.core.view.isVisible
 8  import androidx.fragment.app.Fragment
 9  import androidx.fragment.app.activityViewModels
10  import androidx.lifecycle.lifecycleScope
11  import androidx.navigation.fragment.navArgs
12  import coil.load
13  import
    com.example.dreamstravel.databinding.FragmentDetailBinding
14  import com.example.dreamstravel.viewmodel.CountryViewModel
15  import
    com.example.dreamstravel.viewmodel.CountryViewModelFactory
16  import kotlinx.coroutines.launch
17
18  class DetailFragment : Fragment() {
19
20      private var _binding: FragmentDetailBinding? = null
        private val binding get() = _binding!!
21
22      private val args: DetailFragmentArgs by navArgs()
23
24
```

```
25      private val viewModel: CountryViewModel by
    activityViewModels {
26          CountryViewModelFactory(
27              requireActivity().application,
28              (requireActivity().application as
    DreamsTravelApp).repository
29          )
30      }
31
32      override fun onCreateView(
33          inflater: LayoutInflater, container: ViewGroup?,
34          savedInstanceState: Bundle?
35      ): View {
36          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
37          return binding.root
38      }
39
40      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
41          super.onViewCreated(view, savedInstanceState)
42
43          binding.progressBar.visibility = View.VISIBLE
44          binding.contentGroup.visibility = View.GONE
45
46          viewModel.loadCountryByName(args.name)
47
48          viewLifecycleOwner.lifecycleScope.launch {
49              viewModel.countryDetail.collect { country ->
50                  binding.progressBar.visibility = View.GONE
51
52                  country?.let {
53                      binding.contentGroup.visibility =
    View.VISIBLE
54                      binding.tvDetailName.text = it.nama
55                      binding.tvDetailAlasan.text =
    it.alasan


56
57  binding.imgDetailPhoto.load(it.imageUrl) {

58  placeholder(R.drawable.ic_launcher_background)

59  error(R.drawable.ic_launcher_foreground)
60                      }
61                  }
62              }
```

```
63          }
64      }
65
66      override fun onDestroyView() {
67          super.onDestroyView()
68          _binding = null
69      }
70  }
```

3. HomeFragment.kt

Tabel 29. Source Code Jawaban Soal 1 Modul 5 HomeFragment.kt

```
1   package com.example.dreamstravel
2
3   import android.content.Intent
4   import android.os.Bundle
5   import android.util.Log
6   import android.view.LayoutInflater
7   import android.view.View
8   import android.view.ViewGroup
9   import androidx.core.net.toUri
10  import androidx.core.view.isVisible
11  import androidx.fragment.app.Fragment
12  import androidx.fragment.app.activityViewModels
13  import androidx.lifecycle.lifecycleScope
14  import androidx.navigation.fragment.findNavController
15  import androidx.recyclerview.widget.LinearLayoutManager
16  import com.example.dreamstravel.databinding.FragmentHomeBinding
17  import com.example.dreamstravel.viewmodel.CountryViewModel
18  import
    com.example.dreamstravel.viewmodel.CountryViewModelFactory
19  import kotlinx.coroutines.launch
20
21  class HomeFragment : Fragment() {
22      private var _binding: FragmentHomeBinding? = null
23      private val binding get() = _binding!!
24
25      private     val     viewModel:     CountryViewModel     by
    activityViewModels {
26          CountryViewModelFactory(
27              requireActivity().application,
28              (requireActivity().application                    as
    DreamsTravelApp).repository
29          )
30      }
31      private lateinit var adapter: CountryAdapter
```

```
32
33      override fun onCreateView(inflater: LayoutInflater,
    container: ViewGroup?, savedInstanceState: Bundle?): View {
34          _binding = FragmentHomeBinding.inflate(inflater,
35  container, false)
            return binding.root
36      }
37
38      override fun onViewCreated(view: View, savedInstanceState:
    Bundle?) {
39          super.onViewCreated(view, savedInstanceState)
40          setupRecyclerView()
41
42          lifecycleScope.launch {
43              viewModel.listCountryState.collect { state ->
44                  when (state) {
45                      is UiState.Loading -> {
46                          binding.progressBar.isVisible = true
47                          binding.rvNegara.isVisible = false
48                          binding.tvError.isVisible = false
49                      }
50                      is UiState.Success -> {
51                          binding.progressBar.isVisible = false
52                          binding.rvNegara.isVisible = true
53                          binding.tvError.isVisible = false
54                          adapter.updateData(state.data)
55                      }
56                      is UiState.Error -> {
57                          binding.progressBar.isVisible = false
58                          binding.rvNegara.isVisible = false
59                          binding.tvError.isVisible = true
60                          binding.tvError.text = state.message
61                      }
62                  }
63              }
64          }
65      }
66
67      private fun setupRecyclerView() {
68          adapter = CountryAdapter(
69              countries = emptyList(),
70              onDetailClick = { country ->
71                  country.nama?.let { name ->
72                      val action =
    HomeFragmentDirections.actionHomeFragmentToDetailFragment(name)
73                      findNavController().navigate(action)
74                  }
```

```
75              },
76              onWikiClick = { country ->
77                  val    intent    =    Intent(Intent.ACTION_VIEW,
    country.wikiUrl.toUri())
78                  startActivity(intent)
79              },
80              onFavoriteClick = { country ->
81                  viewModel.toggleFavorite(country)
82              }
83          )
84          binding.rvNegara.layoutManager                              =
    LinearLayoutManager(requireContext())
85          binding.rvNegara.adapter = adapter
86      }
87
88      override fun onDestroyView() {
89          super.onDestroyView()
90          _binding = null
91      }
92  }
```

4. CountryAdapter.kt

Tabel 30. Source Code Jawaban Soal 1 Modul 5 CountryAdapter.kt

```
1   package com.example.dreamstravel
2
3   import android.annotation.SuppressLint
4   import android.view.LayoutInflater
5   import android.view.ViewGroup
6   import androidx.recyclerview.widget.RecyclerView
7   import coil.load
8   import com.example.dreamstravel.data.local.CountryEntity
9   import com.example.dreamstravel.databinding.ItemNegaraBinding
10
11  class CountryAdapter(
12      private var countries: List<CountryEntity>,
13      private val onDetailClick: (CountryEntity) -> Unit,
14      private val onWikiClick: (CountryEntity) -> Unit,
15      private val onFavoriteClick: (CountryEntity) -> Unit
16  ) : RecyclerView.Adapter<CountryAdapter.CountryViewHolder>() {
17
18      inner      class      CountryViewHolder(val      binding:
    ItemNegaraBinding) : RecyclerView.ViewHolder(binding.root)
        override    fun    onCreateViewHolder(parent:    ViewGroup,
19  viewType: Int): CountryViewHolder {
```

```
20          val                         binding                         =
    ItemNegaraBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
21          return CountryViewHolder(binding)
22      }
23
24      override  fun  onBindViewHolder(holder:  CountryViewHolder,
    position: Int) {
25          val country = countries[position]
26          with(holder.binding) {
27              tvItemName.text = country.nama
28              tvAlasan.text = country.alasan
29
30              imgItemPhoto.load(country.imageUrl) {
31                  placeholder(R.drawable.ic_launcher_background)
32                  error(R.drawable.ic_launcher_foreground)
33              }
34
35              buttonDetail.setOnClickListener                    {
    onDetailClick(country) }
36              btnWiki.setOnClickListener  {  onWikiClick(country)
    }
37
38              btnFavorite.setImageResource(
39                  if (country.isFavorite) R.drawable.ic_favorite
40                  else R.drawable.ic_favorite_border
41              )
42              btnFavorite.setOnClickListener                     {
43  onFavoriteClick(country) }
            }
44      }
45
46      override fun getItemCount(): Int = countries.size
47
48      @SuppressLint("NotifyDataSetChanged")
49      fun updateData(newList: List<CountryEntity>) {
50          countries = newList
51          notifyDataSetChanged()
52      }
53  }
```

5. DreamsTravelApp.kt

Tabel 31. Source Code Jawaban Soal 1Modul 5 DreamsTravelApp.kt

```
1   package com.example.dreamstravel
2
```

```
3  import android.app.Application
4  import
   com.example.dreamstravel.data.local.CountryRepository
5  import com.example.dreamstravel.data.local.CountryDatabase
6
7  class DreamsTravelApp : Application() {
8      val database by lazy {
9  CountryDatabase.getInstance(this) }
       val repository by lazy {
10 CountryRepository(database.countryDao()) }
11 }
```

6. FavoriteFragment.kt

Tabel 32. Source Code Jawaban Soal 1 Modul 5 FavoriteFragmentl.kt

```
1  package com.example.dreamstravel
2
3  import android.content.Intent
4  import android.os.Bundle
5  import android.util.Log
6  import android.view.LayoutInflater
7  import android.view.View
8  import android.view.ViewGroup
9  import androidx.core.net.toUri
10 import androidx.core.view.isVisible
11 import androidx.fragment.app.Fragment
12 import androidx.fragment.app.activityViewModels
13 import androidx.lifecycle.lifecycleScope
14 import androidx.navigation.fragment.findNavController
15 import androidx.recyclerview.widget.LinearLayoutManager
16 import
   com.example.dreamstravel.databinding.FragmentFavoriteBinding
17 import com.example.dreamstravel.viewmodel.CountryViewModel
18 import com.example.dreamstravel.viewmodel.CountryViewModelFactory
19 import kotlinx.coroutines.launch
20
21 class FavoriteFragment : Fragment() {
22     private var _binding: FragmentFavoriteBinding? = null
23     private val binding get() = _binding!!
24
25     private val viewModel: CountryViewModel by activityViewModels
26 {
27         CountryViewModelFactory(
28             requireActivity().application,
                (requireActivity().application                    as
29 DreamsTravelApp).repository
```

```
        )
    }
    private lateinit var adapter: CountryAdapter

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

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

        viewModel.loadFavoriteCountries()

        lifecycleScope.launch {
            viewModel.favoriteListState.collect { state ->
                when (state) {
                    is UiState.Loading -> {
                        binding.progressBar.isVisible = true
                        binding.rvNegara.isVisible = false
                        binding.tvInfo.isVisible = false
                    }
                    is UiState.Success -> {
                        binding.progressBar.isVisible = false
                        if (state.data.isEmpty()) {
                            binding.rvNegara.isVisible = false
                            binding.tvInfo.isVisible = true
                            binding.tvInfo.text = "Belum ada negara favorit"
                        } else {
                            binding.rvNegara.isVisible = true
                            binding.tvInfo.isVisible = false
                            adapter.updateData(state.data)
                        }
                    }
                    is UiState.Error -> {
                        binding.progressBar.isVisible = false
                        binding.rvNegara.isVisible = false
                        binding.tvInfo.isVisible = true
                        binding.tvInfo.text = state.message
                    }
                }
            }
```

```
72          }
73      }
74
75      private fun setupRecyclerView() {
76          adapter = CountryAdapter(
77              countries = emptyList(),
78              onDetailClick = { country ->
79
80                  country.nama?.let { name ->
81                      val                  action                  =
    FavoriteFragmentDirections.actionFavoriteFragmentToDetailFragment
82  (name)
                        findNavController().navigate(action)
                    }
83              },
84              onWikiClick = { country ->
85                  val    intent    =    Intent(Intent.ACTION_VIEW,
86  country.wikiUrl.toUri())
87                  startActivity(intent)
                },
88              onFavoriteClick = { country ->
89                  viewModel.toggleFavorite(country)
90              }
91          )
92          binding.rvNegara.layoutManager                              =
93  LinearLayoutManager(requireContext())
94          binding.rvNegara.adapter = adapter
        }
95
96      override fun onDestroyView() {
97          super.onDestroyView()
98          _binding = null
99      }
100 }
101
```

7. UiState.kt

Tabel 33. Source Code Jawaban Soal 1 Modul 5 UiState.kt

```
1   package com.example.dreamstravel
2
3   sealed class UiState<out T> {
4       object Loading : UiState<Nothing>()
5
6       data class Success<out T>(val data: T) : UiState<T>()
7
```

```
8       data class Error(val message: String) :
        UiState<Nothing>()
9   }
```

8. viewmodel/CountryViewModel.kt

Tabel 34. Source Code Jawaban Soal 1 Modul 5 CountryViewModel.kt

```
1   package com.example.dreamstravel.viewmodel
2
3   import android.app.Application
4   import android.util.Log
5   import androidx.lifecycle.AndroidViewModel
6   import androidx.lifecycle.viewModelScope
7   import com.example.dreamstravel.UiState
8   import com.example.dreamstravel.data.local.CountryEntity
9   import com.example.dreamstravel.data.local.CountryRepository
10  import kotlinx.coroutines.flow.MutableStateFlow
11  import kotlinx.coroutines.flow.StateFlow
12  import kotlinx.coroutines.launch
13  import java.io.IOException
14
15  class CountryViewModel(
16      application: Application,
17      private val repository: CountryRepository
18  ) : AndroidViewModel(application) {
19
20      private val _listCountryState =
        MutableStateFlow<UiState<List<CountryEntity>>>(UiState.Loading)
21      val listCountryState:
        StateFlow<UiState<List<CountryEntity>>> get() =
        _listCountryState
22
23      private val _favoriteListState =
        MutableStateFlow<UiState<List<CountryEntity>>>(UiState.Loading)
24      val favoriteListState:
        StateFlow<UiState<List<CountryEntity>>> get() =
        _favoriteListState
25
26      private val _countryDetail =
        MutableStateFlow<CountryEntity?>(null)
27      val countryDetail: StateFlow<CountryEntity?> get() =
        _countryDetail
28
29      init {
30          loadCountries()
31      }
```

```
32
33      fun loadCountries() {
34          _listCountryState.value = UiState.Loading
35          viewModelScope.launch {
36              try {
37
38                  val remoteData =
repository.fetchCountriesFromApi()
39                  repository.insertAll(remoteData) //
Simpan/Update ke database
40
41                  repository.getAllCountries().collect {
countries ->
42                      _listCountryState.value =
UiState.Success(countries)
43                  }
44              } catch (e: IOException) {
45                  _listCountryState.value = UiState.Error("Gagal
terhubung ke server. Periksa koneksi internet.")
46              } catch (e: Exception) {
47                  _listCountryState.value =
UiState.Error("Terjadi kesalahan: ${e.message}")
48              }
49          }
50      }
51      fun loadFavoriteCountries() {
52          _favoriteListState.value = UiState.Loading
53          viewModelScope.launch {
54              try {
55                  repository.getFavoriteCountries().collect {
favorites ->
56                      _favoriteListState.value =
UiState.Success(favorites)
57                  }
58              } catch (e: Exception) {
59                  _favoriteListState.value = UiState.Error("Gagal
memuat daftar favorit: ${e.message}")
60              }
61          }
62      }
63
64      fun loadCountryByName(countryName: String) {
65          viewModelScope.launch {
66              repository.getCountryByName(countryName).collect {
country ->
67                  _countryDetail.value = country
68              }
```

```
69          }
70      }
71
72      fun toggleFavorite(country: CountryEntity) {
73          viewModelScope.launch {
74              val updated = country.copy(isFavorite =
    !country.isFavorite)
75              repository.updateCountry(updated)
76
77              if (_favoriteListState.value is UiState.Success) {
78                  loadFavoriteCountries()
79              }
80          }
81      }
82  }
```

9. viewmodel/CountryViewModelFactory.kt

Tabel 35. Source Code Jawaban Soal 1 Modul 5 viewmodel/CountryViewModelFactory.kt

```
1   package com.example.dreamstravel.viewmodel
2
3   import android.app.Application
4   import androidx.lifecycle.ViewModel
5   import androidx.lifecycle.ViewModelProvider
6   import
    com.example.dreamstravel.data.local.CountryRepository
7
8   class CountryViewModelFactory(
9       private val application: Application,
10      private val repository: CountryRepository
11  ) : ViewModelProvider.Factory {
12
13      override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
14          if
    (modelClass.isAssignableFrom(CountryViewModel::class.java))
    {
15              @Suppress("UNCHECKED_CAST")
16              return CountryViewModel(application,
    repository) as T
17          }
18          throw IllegalArgumentException("Unknown ViewModel
    class")
19      }
20  }
```

10. data/local/AppDataBase.kt

Tabel 36. Source Code Jawaban Soal 1 Modul 5 data/local/AppDataBase.kt

```
1   package com.example.dreamstravel.data.local
2
3   import android.content.Context
4   import androidx.room.Database
5   import androidx.room.Room
6   import androidx.room.RoomDatabase
7
8   @Database(entities = [CountryEntity::class], version = 2,
    exportSchema = false)
9   abstract class CountryDatabase : RoomDatabase() {
10
11      abstract fun countryDao(): CountryDao
12
13      companion object {
14          @Volatile
15          private var INSTANCE: CountryDatabase? = null
16
17          fun getInstance(context: Context): CountryDatabase
    {
18              return INSTANCE ?: synchronized(this) {
19                  val instance = Room.databaseBuilder(

20  context.applicationContext,

21  CountryDatabase::class.java,
22                                      "country_database"

23  ).fallbackToDestructiveMigration(false).build()
24                  INSTANCE = instance
25                  instance
26              }
27          }
28      }
29  }
```

11. data/local/CountryDao.kt

Tabel 37. Source Code Jawaban Soal 1 Modul 5 data/local/CountryDao.kt

```
1   package com.example.dreamstravel.data.local
2
3   import androidx.room.Dao
4   import androidx.room.Insert
5   import androidx.room.OnConflictStrategy
```

```
6  import androidx.room.Query
7  import kotlinx.coroutines.flow.Flow
8  import androidx.room.Update
9
10 @Dao
11 interface CountryDao {
12
13     @Query("SELECT * FROM country")
14     fun getAllCountries(): Flow<List<CountryEntity>>
15
16     @Query("SELECT * FROM country WHERE nama = :name LIMIT
   1")
17     fun           getCountryByName(name:           String):
   Flow<CountryEntity?>
18
19     @Insert(onConflict = OnConflictStrategy.REPLACE)
20     suspend fun insertAll(countries: List<CountryEntity>)
21
22     @Query("DELETE FROM country")
23     suspend fun deleteAll()
24
25     @Query("SELECT * FROM country WHERE isFavorite = 1 ORDER
   BY nama ASC")
26     fun getFavoriteCountries(): Flow<List<CountryEntity>>
27
28     @Update
29     suspend fun updateCountry(country: CountryEntity)
30 }
```

12. data/local/CountryEntity.kt

Tabel 38. Source Code Jawaban Soal 1 Modul 5 data/local/CountryEntity.kt

```
1  package com.example.dreamstravel.data.local
2  import androidx.room.Entity
3  import androidx.room.PrimaryKey
4
5  @Entity(tableName = "country")
6  data class CountryEntity(
7      @PrimaryKey(autoGenerate = true)
8      val id: Int = 0,
9      val nama: String?,
10     val alasan: String,
11     val imageUrl: String?,
12     val wikiUrl: String,
13     val isFavorite: Boolean = false
14 )
```

13. data/local/CountryRepository.kt

Tabel 39. Source Code Jawaban Soal 1 Modul 5 data/local/CountryRepository.kt

```
1   package com.example.dreamstravel.data.local
2
3   import
    com.example.dreamstravel.data.remote.CountryResponse
4   import com.example.dreamstravel.data.remote.RetrofitClient
5   import kotlinx.coroutines.flow.Flow
6
7   class CountryRepository(private val countryDao:
    CountryDao) {
8
9       fun getAllCountries(): Flow<List<CountryEntity>> {
10          return countryDao.getAllCountries()
11      }
12
13      fun getCountryByName(name: String):
    Flow<CountryEntity?> { // Pastikan return type bisa null
14          return countryDao.getCountryByName(name)
15      }
16
17      suspend fun insertAll(countries: List<CountryEntity>)
    {
18          countryDao.insertAll(countries)
19      }
20
21      suspend fun updateCountry(country: CountryEntity) {
22          countryDao.updateCountry(country)
23      }
24
25      fun getFavoriteCountries(): Flow<List<CountryEntity>>
    {
26          return countryDao.getFavoriteCountries()
27      }
28
29      suspend fun fetchCountriesFromApi():
    List<CountryEntity> {
30          val response: List<CountryResponse> =
    RetrofitClient.apiService.getAllCountries()
31
32          return response.mapNotNull { countryResponse ->
33              val countryName = countryResponse.name?.common
34              val flagUrl = countryResponse.flags?.png
35              val region = countryResponse.region
36
37
```

```
            if (countryName.isNullOrEmpty() ||
flagUrl.isNullOrEmpty()) {
                null
            } else {
                CountryEntity(
                    nama = countryName,
                    tahun = "2025",
                    alasan = region ?: "No region",
                    imageUrl = flagUrl,
                    wikiUrl =
"https://en.wikipedia.org/wiki/${countryName.replace(" ",
"_")}"
                )
            }
        }
    }
```

14. data/remote/ApisService.kt

Tabel 40. Source Code Jawaban Soal 1 Modul 5 data/remote/ApiService.kt

```
package com.example.dreamstravel.data.remote

import retrofit2.http.GET
import retrofit2.http.Query

interface ApiService {
    @GET("all")
    suspend fun getAllCountries(
        @Query("fields") fields: String =
"name,flags,region"
    ): List<CountryResponse>
}
```

15. data/remote/CountryResponse.kt

Tabel 41. Source Code Jawaban Soal 1 Modul 5 data/remote/CountryResponse.kt

```
package com.example.dreamstravel.data.remote
import kotlinx.serialization.Serializable

@Serializable
data class CountryResponse(
    val name: Name?,
    val flags: Flags?,
    val region: String?
```

```
)

@Serializable
data class Name(val common: String?)

@Serializable
data class Flags(val png: String?)
```

16. data/remote/RetrofitClient.kt

Tabel 42. Source Code Jawaban Soal 1Modul 5 data/remote/RetrofitClient.kt

```
package com.example.dreamstravel.data.remote

import
com.jakewharton.retrofit2.converter.kotlinx.serialization
.asConverterFactory
import kotlinx.serialization.ExperimentalSerializationApi
import kotlinx.serialization.json.Json
import okhttp3.MediaType.Companion.toMediaType
import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit

@OptIn(ExperimentalSerializationApi::class)
object RetrofitClient {
    private val json = Json { ignoreUnknownKeys = true }

    private val loggingInterceptor =
HttpLoggingInterceptor().apply
 {
        level = HttpLoggingInterceptor.Level.BODY // BODY akan
menampilkan semua detail
    }

    private val client = OkHttpClient.Builder()
        .addInterceptor(loggingInterceptor)
        .build()

    val apiService: ApiService by lazy {
        Retrofit.Builder()
            .baseUrl("https://restcountries.com/v3.1/")
            .client(client)

.addConverterFactory(json.asConverterFactory("application/json"
.toMediaType()))
            .build()
            .create(ApiService::class.java)
```

```
31      }
32  }
```

17. layout/activity_main.xml

Tabel 43. Source Code Jawaban Soal 1Modul 5 layout/activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       tools:context=".MainActivity">
8
9       <androidx.fragment.app.FragmentContainerView
10          android:id="@+id/nav_host_fragment"
11
    android:name="androidx.navigation.fragment.NavHostFragment"
12          android:layout_width="0dp"
13          android:layout_height="0dp"
14          app:defaultNavHost="true"
15
    app:layout_constraintBottom_toTopOf="@id/bottom_navigation"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          app:navGraph="@navigation/nav_graph" />
20
21
    <com.google.android.material.bottomnavigation.BottomNavigationView
22          android:id="@+id/bottom_navigation"
23          android:layout_width="0dp"
24          android:layout_height="wrap_content"
25          app:layout_constraintBottom_toBottomOf="parent"
26          app:layout_constraintEnd_toEndOf="parent"
27          app:layout_constraintStart_toStartOf="parent"
28          app:menu="@menu/bottom_nav_menu"
29          app:itemIconTint="@color/bottom_nav_color_selector"
30          app:itemTextColor="@color/bottom_nav_color_selector" />
31
32  </androidx.constraintlayout.widget.ConstraintLayout>
```

18. layout/fragment_detail.xml

Tabel 44. Source Code Jawaban Soal 1 Modul 5 layout/fragment_detail.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:fillViewport="true">
9
10     <androidx.constraintlayout.widget.ConstraintLayout
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         tools:context=".DetailFragment">
14
15         <ProgressBar
16             android:id="@+id/progressBar"
17             style="?android:attr/progressBarStyle"
18             android:layout_width="wrap_content"
19             android:layout_height="wrap_content"
20             android:visibility="gone"
21             app:layout_constraintBottom_toBottomOf="parent"
22             app:layout_constraintEnd_toEndOf="parent"
23             app:layout_constraintStart_toStartOf="parent"
24             app:layout_constraintTop_toTopOf="parent"
25             tools:visibility="visible" />
26
27         <androidx.constraintlayout.widget.Group
28             android:id="@+id/content_group"
29             android:layout_width="wrap_content"
30             android:layout_height="wrap_content"

31 app:constraint_referenced_ids="imgDetailPhoto,tvDetailName,
   tvDetailAlasan" />
32
33         <ImageView
34             android:id="@+id/imgDetailPhoto"
35             android:layout_width="200dp"
36             android:layout_height="200dp"
37             android:layout_marginTop="32dp"
38             android:contentDescription="@string/img_desc"
39             app:layout_constraintEnd_toEndOf="parent"
40             app:layout_constraintStart_toStartOf="parent"
41             app:layout_constraintTop_toTopOf="parent"
42             tools:src="@tools:sample/avatars" />
```

```
43
44          <TextView
45              android:id="@+id/tvDetailName"
46              android:layout_width="0dp"
47              android:layout_height="wrap_content"
48              android:layout_marginStart="16dp"
49              android:layout_marginTop="16dp"
50              android:layout_marginEnd="16dp"
51              android:textAlignment="center"
52              android:textSize="24sp"
53              android:textStyle="bold"
54              app:layout_constraintEnd_toEndOf="parent"
55              app:layout_constraintStart_toStartOf="parent"
56
    app:layout_constraintTop_toBottomOf="@id/imgDetailPhoto"
57              tools:text="Country Name" />
58
59          <TextView
60              android:id="@+id/tvDetailAlasan"
61              android:layout_width="0dp"
62              android:layout_height="wrap_content"
63              android:layout_margin="16dp"
64              android:textSize="16sp"
65              app:layout_constraintEnd_toEndOf="parent"
66              app:layout_constraintStart_toStartOf="parent"

67  app:layout_constraintTop_toBottomOf="@id/tvDetailName"
68              tools:text="Reason text goes here..." />
69
70      </androidx.constraintlayout.widget.ConstraintLayout>
71  </ScrollView>
```

19. layout/fragment_favorite.xml

Tabel 45. Source Code Jawaban Soal 1Modul 5 layout/fragment_favorite.xml

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

```
11          android:layout_width="0dp"
12          android:layout_height="0dp"
13          android:clipToPadding="false"
14          android:padding="8dp"
15          app:layout_constraintBottom_toBottomOf="parent"
16          app:layout_constraintEnd_toEndOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintTop_toTopOf="parent"
19          tools:listitem="@layout/item_negara" />
20
21      <ProgressBar
22          android:id="@+id/progressBar"
23          style="?android:attr/progressBarStyle"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          android:visibility="gone"
27          app:layout_constraintBottom_toBottomOf="parent"
28          app:layout_constraintEnd_toEndOf="parent"
29          app:layout_constraintStart_toStartOf="parent"
30          app:layout_constraintTop_toTopOf="parent"
31          tools:visibility="visible" />
32
33      <TextView
34          android:id="@+id/tv_info"
35          android:layout_width="wrap_content"
36          android:layout_height="wrap_content"
37          android:text="Belum ada negara favorit"
38          android:visibility="gone"
39          app:layout_constraintBottom_toBottomOf="parent"
40          app:layout_constraintEnd_toEndOf="parent"
41          app:layout_constraintStart_toStartOf="parent"
42          app:layout_constraintTop_toTopOf="parent"
43          tools:visibility="visible" />
44
45  </androidx.constraintlayout.widget.ConstraintLayout>
```

20. layout/fragment_home.xml

Tabel 46. Source Code Jawaban Soal 1 Modul 5 layout/fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       xmlns:tools="http://schemas.android.com/tools">
```

```
7
8       <androidx.recyclerview.widget.RecyclerView
9           android:id="@+id/rvNegara"
10          android:layout_width="0dp"
11          android:layout_height="0dp"
12          android:clipToPadding="false"
13          android:padding="8dp"
14          android:contentDescription="cute"
15          app:layout_constraintTop_toTopOf="parent"
16          app:layout_constraintBottom_toBottomOf="parent"
17          app:layout_constraintStart_toStartOf="parent"
18          app:layout_constraintEnd_toEndOf="parent"
19          tools:listitem="@layout/item_negara"
20          tools:ignore="HardcodedText" />
21
22      <ProgressBar
23          android:id="@+id/progressBar"
24          style="?android:attr/progressBarStyle"
25          android:layout_width="wrap_content"
26          android:layout_height="wrap_content"
27          android:visibility="gone"
28          app:layout_constraintBottom_toBottomOf="parent"
29          app:layout_constraintEnd_toEndOf="parent"
30          app:layout_constraintStart_toStartOf="parent"
31          app:layout_constraintTop_toTopOf="parent"
32          tools:visibility="visible" />
33
34      <TextView
35          android:id="@+id/tv_error"
36          android:layout_width="wrap_content"
37          android:layout_height="wrap_content"
38          android:text="Error Message"
39          android:visibility="gone"
40          app:layout_constraintBottom_toBottomOf="parent"
41          app:layout_constraintEnd_toEndOf="parent"
42          app:layout_constraintStart_toStartOf="parent"
43          app:layout_constraintTop_toTopOf="parent"
44          tools:visibility="visible" />
45  </androidx.constraintlayout.widget.ConstraintLayout>
```

21. item_negara.xml

Tabel 47. Source Code Jawaban Soal 1 Modul 5 item_negara.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       android:layout_margin="8dp"
8       app:cardCornerRadius="12dp"
9       app:cardElevation="4dp">
10
11      <androidx.constraintlayout.widget.ConstraintLayout
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:padding="12dp">
15
16
    <com.google.android.material.imageview.ShapeableImageView
17              android:id="@+id/img_item_photo"
18              android:layout_width="100dp"
19              android:layout_height="100dp"
20              android:scaleType="centerCrop"
21              android:contentDescription="@string/img_desc"
22              app:layout_constraintStart_toStartOf="parent"
23              app:layout_constraintTop_toTopOf="parent"
24              app:layout_constraintBottom_toBottomOf="parent"
    />
25
26          <TextView
27              android:id="@+id/tv_item_name"
28              android:layout_width="0dp"
29              android:layout_height="wrap_content"
30              android:textSize="18sp"
31              android:textStyle="bold"
32              android:textColor="@android:color/black"
33              android:layout_marginStart="8dp"
34
    app:layout_constraintStart_toEndOf="@id/img_item_photo"
35              app:layout_constraintTop_toTopOf="parent"
36              app:layout_constraintEnd_toEndOf="parent"
37              tools:text="Jepang" />
38
39          <TextView
40              android:id="@+id/tv_alasan"
41              android:layout_width="0dp"
42              android:layout_height="wrap_content"
43              android:layout_marginTop="4dp"
44              android:textColor="@android:color/darker_gray"
45              android:textSize="14sp"
46
```

```
47  app:layout_constraintEnd_toEndOf="@id/tv_item_name"
48  app:layout_constraintStart_toStartOf="@id/tv_item_name"
49  app:layout_constraintTop_toBottomOf="@id/tv_item_name"
50              tools:ignore="TextContrastCheck"
51              tools:text="Asia Timur" />
52
53          <Button
54              android:id="@+id/button_detail"
55              android:layout_width="wrap_content"
56              android:layout_height="wrap_content"
57              android:text="@string/btn_detail"
58              android:layout_marginTop="8dp"
59  app:layout_constraintTop_toBottomOf="@id/tv_alasan"
60  app:layout_constraintStart_toStartOf="@id/tv_item_name"
                app:layout_constraintBottom_toBottomOf="parent"
61  />
62
63          <Button
64              android:id="@+id/btn_wiki"
65              android:layout_width="wrap_content"
66              android:layout_height="wrap_content"
67              android:text="@string/btn_wiki"
68              android:layout_marginStart="8dp"
69              android:layout_marginTop="8dp"
70  app:layout_constraintTop_toBottomOf="@id/tv_alasan"
71  app:layout_constraintStart_toEndOf="@id/button_detail"
                app:layout_constraintBottom_toBottomOf="parent"
72  />
73
74          <ImageButton
75              android:id="@+id/btn_favorite"
76              android:layout_width="48dp"
77              android:layout_height="48dp"
78  android:background="?attr/selectableItemBackgroundBorderless"
79              android:contentDescription="@string/favorite"
80              android:tint="@android:color/holo_red_dark"
81              android:padding="8dp"
82              android:src="@drawable/ic_favorite_border"
83              app:layout_constraintEnd_toEndOf="parent"
```

```
84              app:layout_constraintTop_toTopOf="parent"
85              tools:ignore="UseAppTint" />
86
87      </androidx.constraintlayout.widget.ConstraintLayout>
88  </androidx.cardview.widget.CardView>
```

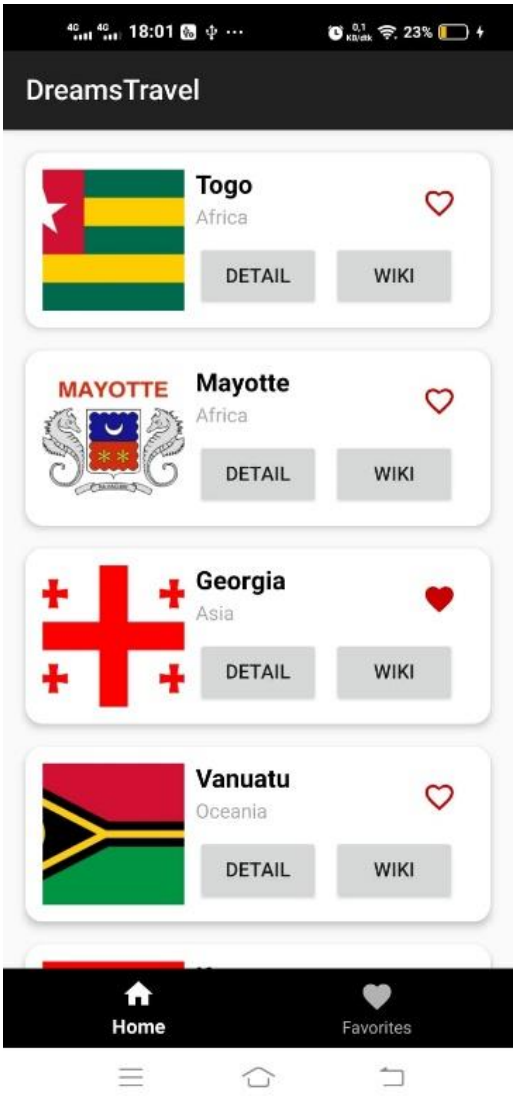
22. navigation/nav_graph.xml

Tabel 48. Source Code Jawaban Soal 1 Modul 5 navigation/nav_graph.xml

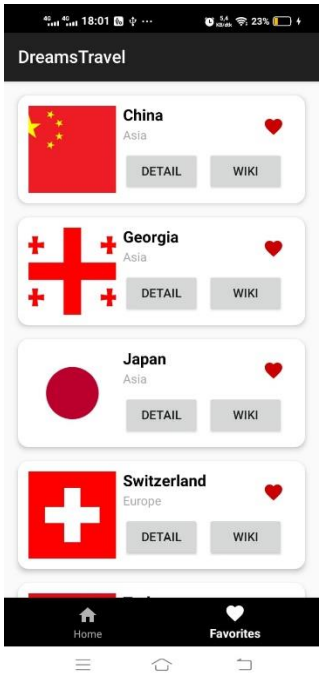
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <navigation
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:id="@+id/nav_graph"
6       app:startDestination="@id/homeFragment">
7
8       <fragment
9           android:id="@+id/homeFragment"
10
    android:name="com.example.dreamstravel.HomeFragment"
11          android:label="Home"
12          tools:layout="@layout/fragment_home">
13          <action
14
    android:id="@+id/action_homeFragment_to_detailFragment"
15              app:destination="@id/detailFragment" />
16      </fragment>
17      <fragment
18          android:id="@+id/favoriteFragment"
19
    android:name="com.example.dreamstravel.FavoriteFragment"
20          android:label="Favorites"
21          tools:layout="@layout/fragment_favorite">
22          <action
23
    android:id="@+id/action_favoriteFragment_to_detailFragment"
24              app:destination="@id/detailFragment" />
25      </fragment>
26      <fragment
27          android:id="@+id/detailFragment"
28
    android:name="com.example.dreamstravel.DetailFragment"
29          android:label="Detail"
30          tools:layout="@layout/fragment_detail">
31          <argument
```

```
32              android:name="name"
33              app:argType="string" />
34      </fragment>
35  </navigation>
```

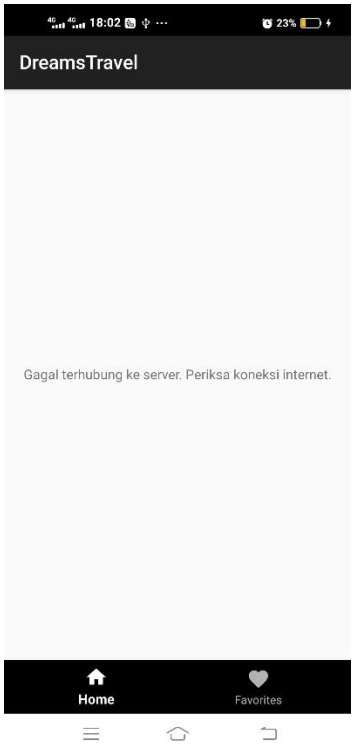
B. Output Program



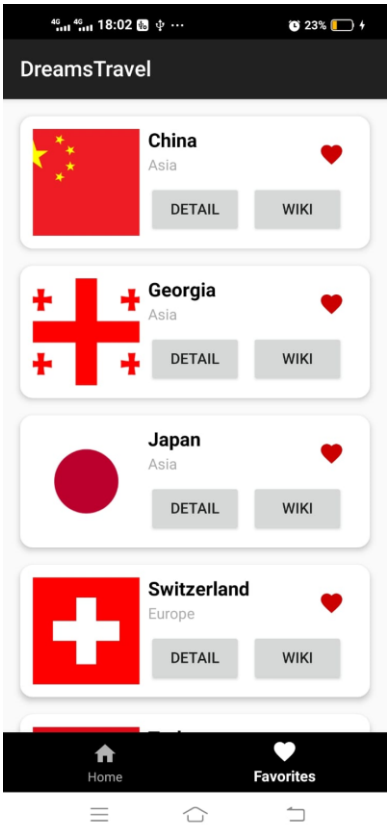
Gambar 15. Tampilan Home



Gambar 16. Tampilan Favorites



Gambar 17. Tampilan Home saat tidak ada jaringan



Gambar 18. Tampilan Favorites saat tidak ada jaringan, masih bisa diakses

Gambar 19. Tampilan Detail

Gambar 20. Tampilan Wiki

C. Pembahasan

1. MainActivity.kt:

Kode di atas adalah bagian dari aplikasi Android yang menggunakan bahasa Kotlin. Kelas `MainActivity` mewarisi `AppCompatActivity` dan berfungsi sebagai aktivitas utama aplikasi. Di dalam metode onCreate, layout utama diatur menggunakan View `Binding (ActivityMainBinding)`. Kemudian, aplikasi mengambil `NavHostFragment` dari layout yang berfungsi sebagai wadah navigasi antar fragment. Setelah mendapatkan NavController dari fragment tersebut, `bottomNavigation` dihubungkan dengan sistem navigasi menggunakan `setupWithNavController,` sehingga menu navigasi bawah dapat digunakan untuk berpindah antar halaman (fragment) dalam aplikasi.

2. DetailFragment.kt

Kode di atas adalah DetailFragment, yaitu salah satu tampilan halaman dalam aplikasi Android yang menampilkan detail sebuah negara. Fragment ini menggunakan View Binding (FragmentDetailBinding) untuk mengakses elemen-elemen UI. Data negara yang akan ditampilkan diterima melalui args (menggunakan navArgs). ViewModel (CountryViewModel) digunakan untuk mengelola dan mengambil data, yang dihubungkan ke Activity dengan bantuan CountryViewModelFactory. Saat tampilan dibuat, fragment menampilkan ProgressBar sambil mengambil data berdasarkan nama negara. Setelah data berhasil dimuat, progress bar disembunyikan, dan data seperti nama, alasan, serta gambar negara ditampilkan di layar. Gambar dimuat menggunakan library Coil. View Binding dihapus saat tampilan dihancurkan untuk menghindari memory leak.

3. **HomeFragment.kt**

   Kode di atas adalah HomeFragment, yaitu tampilan utama aplikasi yang menampilkan daftar negara menggunakan RecyclerView. Fragment ini memanfaatkan View Binding (FragmentHomeBinding) dan CountryViewModel untuk mengambil data. Saat tampilan dibuat, fragment mempersiapkan RecyclerView dan memantau perubahan data dari ViewModel menggunakan collect. Jika data sedang dimuat, maka akan ditampilkan ProgressBar; jika berhasil, daftar negara akan muncul; dan jika gagal, akan tampil pesan error. Adapter memiliki tiga aksi: melihat detail negara (navigasi ke DetailFragment), membuka Wikipedia negara, dan menandai sebagai favorit. Binding dibersihkan saat tampilan dihancurkan untuk mencegah kebocoran memori.

4. **CountryAdapter.kt**

   Kode di atas adalah `CountryAdapter`, yaitu kelas adaptor untuk `RecyclerView` yang menampilkan daftar negara dalam aplikasi. Setiap item dalam daftar menggunakan layout `item_negara.xml` yang diakses melalui View Binding. Adaptor ini menerima daftar negara (`countries`) dan tiga fungsi callback: untuk melihat detail, membuka Wikipedia, dan menandai favorit. Di dalam `onBindViewHolder`, data negara seperti nama, alasan, dan gambar ditampilkan, lalu tombol-tombol diberi aksi yang sesuai. Gambar dimuat menggunakan library `Coil`, dan ikon favorit

berubah tergantung statusnya. Metode `updateData` digunakan untuk memperbarui data daftar negara dan menyegarkan tampilan.

5. **DreamsTravelApp.kt**

   Kode di atas adalah kelas `DreamsTravelApp` yang merupakan turunan dari `Application` dan berfungsi sebagai titik awal global untuk mengelola data di aplikasi. Di dalamnya, objek `database` dan `repository` dibuat secara **lazy** (hanya dibuat saat dibutuhkan). `database` adalah instance dari `CountryDatabase`, dan `repository` adalah `CountryRepository` yang mengambil data dari DAO (`countryDao`). Dengan cara ini, database dan repository bisa diakses dari bagian lain aplikasi secara efisien dan terpusat.

6. **FavoriteFragment.kt**

   Kode di atas adalah `FavoriteFragment`, yaitu tampilan yang menampilkan daftar negara favorit pengguna dalam aplikasi. Fragment ini menggunakan View Binding (`FragmentFavoriteBinding`) dan `CountryViewModel` untuk mengambil data dari database. Saat tampilan dibuat, fragment memanggil `loadFavoriteCountries()` untuk memuat data favorit, lalu mengamati perubahan datanya menggunakan `collect`. Jika sedang memuat, akan tampil `ProgressBar`; jika sukses dan data kosong, akan muncul pesan bahwa belum ada negara favorit; jika sukses dan data ada, daftar negara ditampilkan; jika gagal, akan muncul pesan error. `RecyclerView` menampilkan daftar negara favorit, dengan tiga aksi: lihat detail, buka Wikipedia, dan hapus dari favorit. Binding dibersihkan saat tampilan dihancurkan untuk mencegah memory leak.

7. **UiState.kt**

   Kode di atas adalah class `UiState`, sebuah **sealed class** yang digunakan untuk merepresentasikan tiga kondisi umum dalam pengambilan data di aplikasi, yaitu: `Loading`, `Success`, dan `Error`. `Loading` menandakan data sedang dimuat, `Success` menampung data yang berhasil diambil, dan `Error` menyimpan pesan kesalahan jika terjadi kegagalan. Dengan menggunakan `UiState`, alur logika aplikasi jadi lebih terstruktur dan mudah dikelola saat menampilkan status data ke pengguna.

8. **viewmodel/CountryViewModel.kt**

Kode di atas adalah `CountryViewModel`, kelas yang berfungsi sebagai jembatan antara UI dan data dalam arsitektur MVVM. ViewModel ini mengelola data negara yang diambil dari `CountryRepository`. Terdapat tiga `StateFlow` untuk memantau status: `listCountryState` untuk semua negara, `favoriteListState` untuk negara favorit, dan `countryDetail` untuk detail satu negara. Saat ViewModel dibuat (`init`), data negara dimuat dari API dan disimpan ke database lokal. Fungsi `loadCountries()`, `loadFavoriteCountries()`, dan `loadCountryByName()` digunakan untuk mengambil data sesuai kebutuhan. Fungsi `toggleFavorite()` digunakan untuk menandai atau menghapus negara dari favorit, lalu memperbarui daftar favorit jika perlu. Semua proses berjalan secara asinkron dengan `viewModelScope` menggunakan coroutine.

9. **viewmodel/CountryViewModelFactory.kt**

   Kode di atas adalah `CountryViewModelFactory`, sebuah kelas yang digunakan untuk membuat instance dari `CountryViewModel` dengan parameter tambahan, yaitu `Application` dan `CountryRepository`. Karena `ViewModel` bawaan Android tidak mendukung konstruktor dengan parameter, maka dibutuhkan `ViewModelProvider.Factory` untuk menyediakannya. Dalam metode `create`, jika `modelClass` cocok dengan `CountryViewModel`, maka objek tersebut dibuat dan dikembalikan. Jika tidak cocok, maka akan dilemparkan exception. Factory ini memungkinkan ViewModel digunakan dengan dependensi yang dibutuhkan secara benar dalam arsitektur MVVM.

10. **data/local/AppDataBase.kt**

    Kode di atas adalah `CountryDatabase`, yaitu kelas abstrak yang mewakili database lokal menggunakan Room di Android. Kelas ini mendefinisikan satu entitas `CountryEntity` dan satu DAO `countryDao()` untuk mengakses data. Di dalamnya terdapat `companion object` untuk membuat database sebagai singleton, sehingga hanya ada satu instance database yang aktif selama aplikasi berjalan. Fungsi `getInstance()` memastikan database dibuat sekali dan bisa diakses dari berbagai tempat. `fallbackToDestructiveMigration(false)` berarti jika versi database berubah dan tidak ada migrasi yang disediakan, maka akan terjadi error (tidak otomatis hapus data).

11. data/local/CountryDao.kt

Kode di atas adalah `CountryDao`, yaitu interface yang berisi perintah-perintah untuk mengakses database Room secara langsung. DAO ini mengelola data dari tabel `country`. Fungsi `getAllCountries()` mengambil semua data negara secara real-time menggunakan `Flow`. `getCountryByName(name)` mengambil satu data negara berdasarkan nama. `insertAll()` digunakan untuk menyimpan daftar negara ke database, dan akan mengganti data lama jika ada konflik. `deleteAll()` menghapus semua data negara. `getFavoriteCountries()` mengambil daftar negara yang ditandai sebagai favorit. Terakhir, `updateCountry()` memperbarui data suatu negara di database. Semua operasi berjalan dengan coroutine (`suspend`), sehingga tidak mengganggu UI.

12. data/local/CountryEntity.kt

Kode di atas adalah data class `CountryEntity` yang digunakan sebagai representasi sebuah tabel bernama `country` di database lokal Room. Setiap objek dari kelas ini mewakili satu baris data negara. Properti `id` adalah primary key yang nilainya dibuat otomatis. `nama`, `tahun`, `alasan`, `imageUrl`, dan `wikiUrl` menyimpan informasi tentang nama negara, tahun dikunjungi atau diinginkan, alasan, gambar, dan link Wikipedia. Properti `isFavorite` adalah penanda apakah negara tersebut ditandai sebagai favorit atau tidak. Kelas ini menjadi model utama untuk menyimpan dan mengambil data negara dalam aplikasi.

13. data/local/CountryRepository.kt

Kode di atas adalah implementasi dari `CountryRepository`, yaitu class yang bertanggung jawab untuk mengelola data negara dari dua sumber: database lokal (`CountryDao`) dan API online (`RetrofitClient`). Method `getAllCountries`, `getCountryByName`, dan `getFavoriteCountries` digunakan untuk mengambil data dari database secara real-time (menggunakan Flow). Method `insertAll` dan `updateCountry` digunakan untuk menyimpan dan memperbarui data negara di database. Method `fetchCountriesFromApi` mengambil data negara dari API, memetakan respon tersebut menjadi objek `CountryEntity`, lalu menyiapkan datanya agar bisa disimpan ke database. Kode ini menjembatani sumber data luar dan lokal agar bisa diakses oleh ViewModel.

14. data/remote/ApisService.kt

Kode di atas adalah interface ApiService yang digunakan untuk mendefinisikan endpoint API menggunakan Retrofit. Method getAllCountries() akan melakukan request GET ke endpoint all dan mengambil data negara dari internet. Parameter fields digunakan untuk membatasi data yang diambil, yaitu hanya name, flags, dan region, agar response lebih ringan. Method ini bersifat suspend, artinya harus dipanggil di dalam coroutine karena merupakan operasi jaringan yang bersifat asynchronous (tidak dilakukan di main thread). Data yang diterima akan berupa daftar objek CountryResponse.

15. data/remote/CountryResponse.kt

Kode ini berisi model data untuk membaca respons dari API tentang data negara. Kelas `CountryResponse` mewakili data utama yang berisi tiga properti: `name` (objek `Name?`), `flags` (objek `Flags?`), dan `region` (nama wilayah). Kelas `Name` hanya memiliki satu properti `common` yang berisi nama umum negara, sedangkan kelas `Flags` menyimpan URL gambar bendera dalam format PNG. Semua kelas ditandai dengan `@Serializable`, artinya data ini bisa diubah otomatis dari JSON ke objek Kotlin saat mengambil data dari internet menggunakan Kotlin Serialization.

16. data/remote/RetrofitClient.kt

Kode ini adalah konfigurasi `RetrofitClient` yang digunakan untuk mengambil data negara dari internet menggunakan Retrofit. Di dalamnya, `Json` disiapkan agar dapat mengabaikan data yang tidak dikenal dari respons API. Kemudian, `HttpLoggingInterceptor` ditambahkan untuk mencatat semua detail permintaan dan respons HTTP ke log, yang membantu saat debugging. `OkHttpClient` digunakan sebagai client untuk Retrofit. Lalu, `Retrofit` disiapkan dengan base URL `https://restcountries.com/v3.1/`, client, dan converter `Kotlinx Serialization` untuk mengubah data JSON menjadi objek Kotlin. Objek `apiService` digunakan untuk memanggil endpoint API yang didefinisikan di `ApiService`.

17. layout/activity_main.xml

Kode XML ini adalah layout utama untuk aplikasi Android yang menggunakan ConstraintLayout sebagai wadah utama. Di dalamnya terdapat dua komponen utama:

a. FragmentContainerView dengan ID `nav_host_fragment`, berfungsi sebagai tempat untuk menampilkan fragment yang dikelola oleh Navigation Component. Ini memungkinkan navigasi antar layar (fragment) dalam aplikasi.
b. BottomNavigationView dengan ID `bottom_navigation`, adalah menu navigasi di bagian bawah layar yang memungkinkan pengguna berpindah antar halaman utama aplikasi dengan ikon dan teks.

Layout ini mengatur agar FragmentContainerView memenuhi ruang dari atas hingga tepat di atas BottomNavigationView, sementara BottomNavigationView menempel di bagian bawah layar. Atribut seperti `app:navGraph` dan `app:menu` menghubungkan ke file navigasi dan menu masing-masing.

18. layout/fragment_detail.xml

Kode XML ini merupakan desain tampilan detail pada aplikasi Android yang dibungkus dengan ScrollView agar kontennya bisa digulir jika melebihi ukuran layar. Di dalamnya terdapat ConstraintLayout sebagai wadah utama yang menyusun beberapa elemen UI. Pertama, terdapat ProgressBar yang berguna untuk menampilkan indikator loading saat data sedang dimuat, namun secara default disembunyikan. Selanjutnya, terdapat Group yang mengelompokkan tiga elemen utama: gambar, nama, dan alasan, sehingga bisa dikontrol tampilannya secara bersamaan. Elemen ImageView digunakan untuk menampilkan gambar, sedangkan dua TextView masing-masing menampilkan nama secara mencolok dan alasan dalam bentuk paragraf. Semua elemen diatur secara responsif di tengah layar dengan margin yang cukup agar tampilan tetap rapi dan mudah dibaca.

19. layout/fragment_favorite.xml

Kode XML ini merupakan tampilan untuk halaman favorit dalam aplikasi Android, yang menggunakan ConstraintLayout sebagai wadah utama. Di dalamnya terdapat tiga elemen penting. Pertama, RecyclerView dengan ID `rvNegara` digunakan untuk menampilkan daftar negara favorit secara scrollable, dan setiap itemnya mengikuti layout dari `item_negara`. Kedua, ada ProgressBar yang ditampilkan saat data sedang dimuat, namun secara default disembunyikan. Ketiga, terdapat TextView dengan pesan “Belum ada negara favorit” yang akan muncul jika daftar favorit kosong.

Semua elemen diposisikan di tengah layar menggunakan constraint, dan disiapkan agar tampilannya responsif dan informatif.

20. layout/fragment_home.xml

Kode XML ini adalah layout untuk halaman yang menampilkan daftar negara dalam aplikasi Android, menggunakan ConstraintLayout sebagai wadah utama. Di dalamnya terdapat tiga komponen utama. Pertama, RecyclerView dengan ID rvNegara yang digunakan untuk menampilkan daftar negara secara scrollable dan rapi, dengan padding 8dp dan setiap item-nya mengikuti layout item_negara. Kedua, ProgressBar yang berada di tengah layar dan hanya akan muncul saat data sedang dimuat. Ketiga, TextView dengan ID tv_error yang digunakan untuk menampilkan pesan kesalahan jika terjadi error saat memuat data, namun disembunyikan secara default. Semua elemen diposisikan di tengah dan memenuhi layar dengan constraint yang menghubungkan ke parent.

21. item_negara.xml

Kode XML ini merupakan layout untuk satu item dalam daftar (list) yang ditampilkan menggunakan CardView agar terlihat seperti kartu dengan sudut membulat dan bayangan (elevation). Di dalam CardView terdapat ConstraintLayout yang menyusun elemen-elemen secara rapi. Komponen utama di dalamnya adalah ShapeableImageView untuk menampilkan foto negara, diikuti dua TextView untuk menampilkan nama negara dan alasannya. Lalu ada dua Button: satu untuk melihat detail dan satu lagi untuk membuka Wikipedia. Terakhir, ada ImageButton di pojok kanan atas untuk memberi tanda favorit, menggunakan ikon hati. Semua elemen diatur agar tampil sejajar dan responsif dengan tampilan yang bersih dan mudah dibaca.

22. navigation/nav_graph.xml

Kode XML ini adalah file navigasi untuk aplikasi Android yang menggunakan Navigation Component. Di dalamnya didefinisikan tiga fragment utama: HomeFragment, FavoriteFragment, dan DetailFragment. Fragment awal yang akan ditampilkan saat aplikasi dibuka adalah HomeFragment, karena ditentukan sebagai startDestination. Dari HomeFragment dan FavoriteFragment, pengguna bisa berpindah ke DetailFragment melalui aksi (action) navigasi yang sudah ditentukan.

DetailFragment juga menerima argumen bernama "name" dengan tipe data string, yang biasanya digunakan untuk menampilkan detail berdasarkan item yang dipilih. Navigasi ini memungkinkan perpindahan antar layar (fragment) secara terstruktur dan mudah dikontrol.

D. Tautan Git

Berikut adalah tautan untuk semua source code yang telah dibuat.
https://github.com/annacoded/Pemrograman-Mobile