LAPORAN AKHIR PRAKTIKUM
PEMROGRAMAN MOBILE

Oleh:

Allano Lintang Ertantora **NIM. 2310817210004**

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS LAMBUNG MANGKURAT

JUNI 2025

LEMBAR PENGESAHAN
LAPORAN AKHIR PRAKTIKUM PEMROGRAMAN MOBILE

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

Nama Praktikan : Allano Lintang Ertantora
NIM : 2310817210004

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

DAFTAR ISI

DAFTAR GAMBAR

Modul 1: Android Basic With Kotlin

Modul 2: Android Layout

Modul 3: Build A Scrollable List

Modul 4: Viewmodel And Debugging

Modul 5: Connect To The Internet

DAFTAR TABEL

Modul 1: Android Basic With Kotlin

Modul 2: Android Layout

Modul 3: Build A Scrollable List

Modul 4: Viewmodel And Debugging

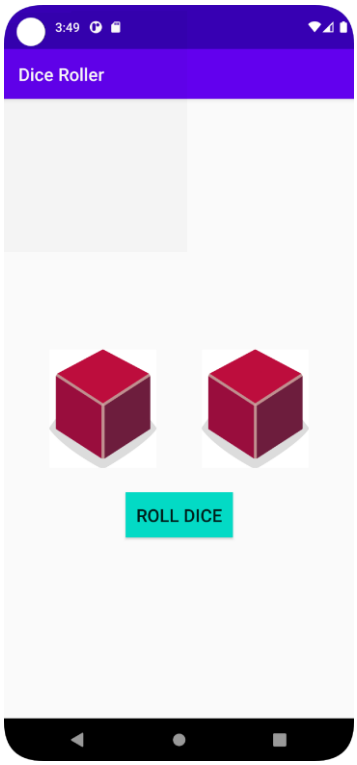
Modul 5: Connect To The Internet

MODUL 1 : ANDROID BASIC WITH KOTLIN

SOAL 1

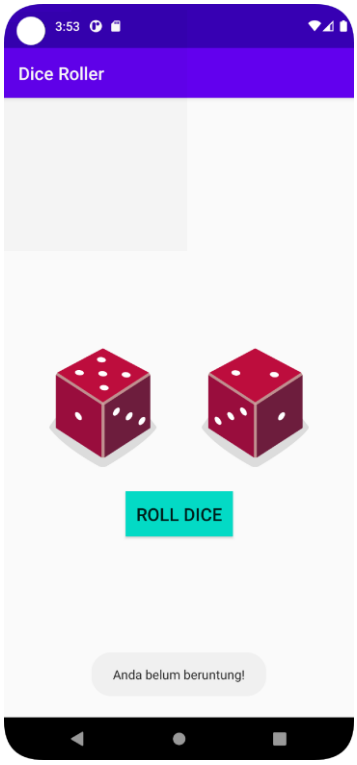
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



Gambar 1 Tampilan Awal Aplikasi

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

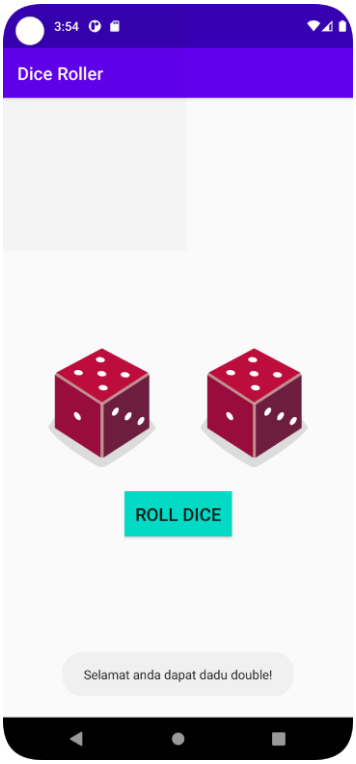


Gambar 2 Tampilan Dadu Setelah Di Roll

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

4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 2 dalam bentuk project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.

5. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/u/0/uc?id=147HT2lIH5qin3z5ta7H9y2N_5OMW81Ll&export= download



Gambar 3 Tampilan Roll Dadu Double

A. Source Code

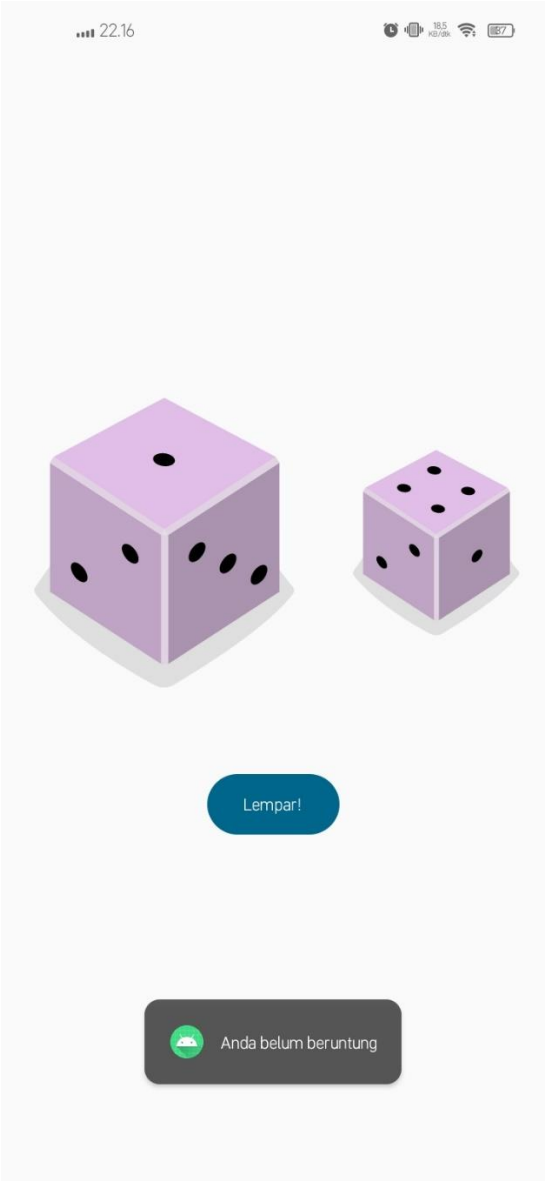
Tabel 1. Source Code Jawaban Soal 1 Modul 1

```
1   package com.example.mobileapptest
2
3   import android.widget.Toast
4   import androidx.compose.ui.platform.LocalContext
5   import android.os.Bundle
6   import androidx.activity.ComponentActivity
7   import androidx.activity.compose.setContent
8   import androidx.activity.enableEdgeToEdge
9   import androidx.compose.foundation.layout.Column
10  import androidx.compose.foundation.layout.Row
11  import androidx.compose.foundation.layout.Spacer
12  import androidx.compose.foundation.layout.fillMaxSize
13  import androidx.compose.foundation.layout.height
14  import androidx.compose.foundation.layout.padding
15  import androidx.compose.foundation.layout.wrapContentSize
16  import androidx.compose.material3.Button
17  import androidx.compose.material3.Scaffold
18  import androidx.compose.material3.Text
19  import androidx.compose.runtime.Composable
20  import androidx.compose.runtime.getValue
21  import androidx.compose.runtime.mutableStateOf
22  import androidx.compose.runtime.remember
```

```
23  import androidx.compose.runtime.setValue
24  import androidx.compose.ui.Alignment
25  import androidx.compose.ui.Modifier
26  import androidx.compose.ui.res.painterResource
27  import androidx.compose.ui.res.stringResource
28  import androidx.compose.ui.tooling.preview.Preview
29  import androidx.compose.ui.unit.dp
30  import com.example.mobileapptest.ui.theme.MobileAppTestTheme
31  import androidx.compose.foundation.Image as Image2
32
33  class MainActivity : ComponentActivity() {
34      override fun onCreate(savedInstanceState: Bundle?) {
35          super.onCreate(savedInstanceState)
36          enableEdgeToEdge()
37          setContent {
38              MobileAppTestTheme {
39                  DiceRollerApp()
40
41              }
42          }
43      }
44  }
45
46  @Preview
47  @Composable
48  fun DiceRollerApp() {
49      DiceButtonWithImage()
50  }
51
52
53  @Composable
54  fun   DiceButtonWithImage(modifier:   Modifier   =   Modifier
    .fillMaxSize() .wrapContentSize(Alignment.Center)) {
55      var result1 by remember{mutableStateOf( 0)}
56      var result2 by remember{mutableStateOf( 0)}
57      val context = LocalContext.current
58
59      val result1Image = getImage(result1)
60      val result2Image = getImage(result2)
61
62      Column (
63          modifier = modifier,
64          horizontalAlignment = Alignment.CenterHorizontally
65      ) {
66          Row(verticalAlignment   =   Alignment.CenterVertically,
    modifier = Modifier){
67              Image2(
68                  painter = painterResource(id = result1Image),
69                  contentDescription = result1.toString(),
70                  modifier = Modifier.padding(8.dp)
71              )
72              Image2(
```

```
73                  painter = painterResource(id = result2Image),
74                  contentDescription = result2.toString(),
75                  modifier = Modifier.padding(8.dp)
76
77              )
78          }
79          Spacer(modifier = Modifier.height(16.dp))
80          Button(onClick = {
81              result1 = (1..6).random()
82              result2 = (1..6).random()
83          if(result1 == result2){
84              Toast.makeText(context, "Selamat anda dapat dadu
    double!", Toast.LENGTH_SHORT).show()
85          }else{
86              Toast.makeText(context,  "Anda  belum  beruntung",
    Toast.LENGTH_SHORT).show()
87          }
88          }) {
89              Text(stringResource(R.string.roll))
90          }
91      }
92
93  }
94
95  @Composable
96  fun getImage(diceValue: Int): Int {
97      return when (diceValue){
98          1 -> R.drawable.dice_1
99          2 -> R.drawable.dice_2
100         3 -> R.drawable.dice_3
101         4 -> R.drawable.dice_4
102         5 -> R.drawable.dice_5
103         6 -> R.drawable.dice_6
104         else -> R.drawable.dice_0
105     }
106 }}
```

B. Output Program



Gambar 1. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

Pada line 1, dideklarasikan package com.example.mobileapptest sebagai identitas unik dari aplikasi.

Pada line 3, diimport Toast untuk menampilkan notifikasi singkat dalam bentuk pop-up.

Pada line 4, diimport LocalContext untuk mendapatkan konteks aplikasi di dalam komponen Compose.

Pada line 5, diimport Bundle untuk menyimpan dan mengirim data saat lifecycle onCreate.

Pada line 6, diimport ComponentActivity sebagai superclass utama dari MainActivity.

Pada line 7, diimport setContent untuk menampilkan antarmuka pengguna menggunakan Jetpack Compose.

Pada line 8, diimport enableEdgeToEdge agar UI dapat menutupi seluruh area layar.

Pada lines 9–15, diimport berbagai komponen layout seperti Column, Row, Spacer, fillMaxSize, height, padding, dan wrapContentSize untuk mengatur tata letak UI.

Pada lines 16–18, diimport Button, Scaffold, dan Text sebagai elemen UI dari Material 3.

Pada lines 19–23, diimport elemen state Compose seperti @Composable, getValue, mutableStateOf, remember, dan setValue untuk mendukung state management.

Pada lines 26–29, diimport elemen UI tambahan seperti Modifier, painterResource, stringResource, Preview, dan unit dp untuk styling.

Pada line 30, diimport MobileAppTestTheme untuk menerapkan tema khusus aplikasi.

Pada line 31, diimport Image dari foundation dan diberi alias Image2 untuk menghindari konflik nama.

Pada line 33, dideklarasikan kelas MainActivity yang merupakan turunan dari ComponentActivity.

Pada line 34, didefinisikan fungsi onCreate() yang akan dijalankan saat aktivitas dibuat.

Pada line 35, dipanggil super.onCreate() untuk mewarisi logika dasar dari superclass.

Pada line 36, dipanggil enableEdgeToEdge() untuk mengizinkan konten menjangkau batas layar.

Pada line 37, dipanggil setContent {} sebagai awal untuk membangun UI dengan Compose.

Pada line 38, diterapkan tema MobileAppTestTheme agar gaya visual aplikasi konsisten.

Pada line 39, dipanggil fungsi DiceRollerApp() untuk menampilkan UI utama aplikasi.

Pada line 46, digunakan anotasi @Preview untuk menampilkan preview UI di Android Studio.

Pada line 48, didefinisikan fungsi DiceRollerApp() sebagai titik masuk UI, yang memanggil DiceButtonWithImage().

Pada line 54, didefinisikan fungsi composable DiceButtonWithImage() dengan modifier default untuk memenuhi dan memusatkan tampilan.

Pada lines 55–56, dideklarasikan dua variabel state result1 dan result2 dengan nilai awal 0, mewakili hasil dua dadu.

Pada line 57, diambil konteks aplikasi melalui LocalContext.current untuk keperluan Toast.

Pada lines 59–60, dipanggil fungsi getImage() untuk mendapatkan ID gambar berdasarkan nilai result1 dan result2.

Pada line 62, digunakan Column untuk menampilkan komponen secara vertikal.

Pada line 63, modifier diterapkan dan konten disejajarkan ke tengah secara horizontal.

Pada line 66, digunakan Row untuk menampilkan gambar dadu secara horizontal.

Pada lines 67–71, ditampilkan gambar result1 dengan padding 8dp.

Pada lines 72–77, ditampilkan gambar result2 juga dengan padding 8dp.

Pada line 79, ditambahkan Spacer dengan tinggi 16dp sebagai jarak antar elemen.

Pada line 80, dideklarasikan tombol Button dengan logika onClick.

Pada lines 81–82, nilai acak 1–6 diberikan ke result1 dan result2.

Pada lines 83–84, jika nilai dadu sama, ditampilkan pesan “Selamat anda dapat dadu double!” menggunakan Toast.

Pada lines 85–86, jika tidak sama, ditampilkan pesan “Anda belum beruntung”.

Pada line 89, digunakan Text() untuk menampilkan teks tombol dengan string dari R.string.roll.

Pada line 96, didefinisikan fungsi getImage() dengan parameter diceValue bertipe Int.

Pada lines 98–103, digunakan when untuk mencocokkan nilai dadu dengan gambar yang sesuai (dice_1 sampai dice_6).

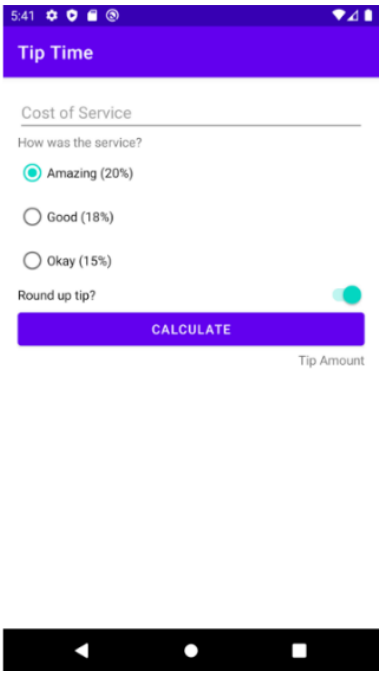
Pada line 104, jika nilai tidak sesuai 1–6, dikembalikan gambar default dice_0

MODUL 2 : ANDROID LAYOUT

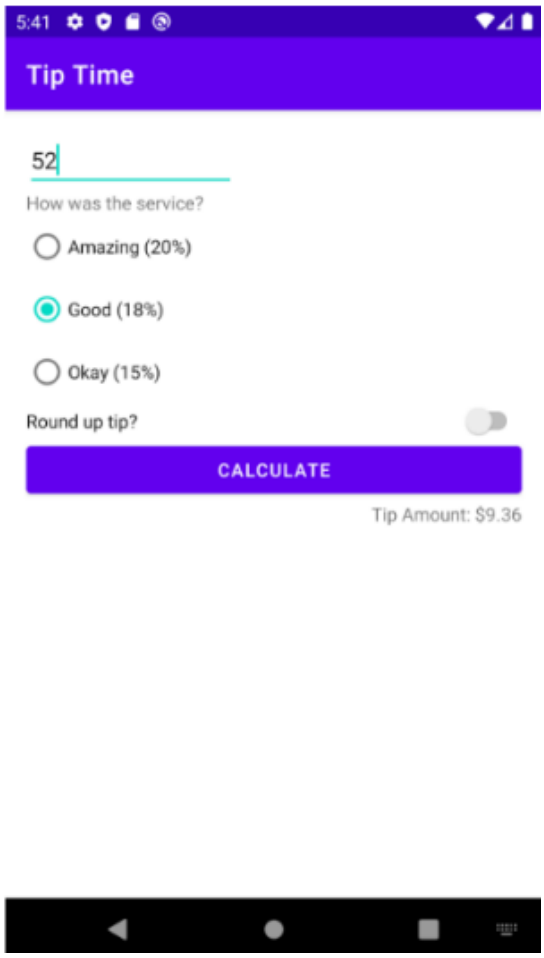
SOAL 1

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

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

```
14 import android.widget.Toast
15 import
   com.example.tipcalc.databinding.ActivityMainBinding
16 import com.example.tipcalc.ui.theme.TipcalcTheme
17
18 class MainActivity : ComponentActivity() {
19
20     lateinit var binding: ActivityMainBinding
21
22     override fun onCreate(savedInstanceState: Bundle?) {
23         super.onCreate(savedInstanceState)
24         binding =
   ActivityMainBinding.inflate(layoutInflater)
25         setContentView(binding.root)
26         binding.calculate.setOnClickListener {
27             hitung()
28         }
29     }
30     private fun hitung(){
31         val cost =
   binding.costOfService.text.toString().toDouble()
32         val tip = binding.tipOptions.checkedRadioButtonId
33         val tipPercent = when (tip) {
34             R.id.option_twenty -> 0.20
35             R.id.option_eighteen -> 0.18
36             else -> 0.15
37         }
38
39         var total = tipPercent * cost
40         val roundUp = binding.roundUp.isChecked
41         if(roundUp){
42             total = kotlin.math.ceil(total)
43         }
44         if(cost <= 0){
45             Toast.makeText(this, "Masukkan nilai yang
   benar", Toast.LENGTH_SHORT).show()
46         }else{
47             binding.total.text = total.toString()
48         }
49     }
50 }
```

2. activity_main.xml

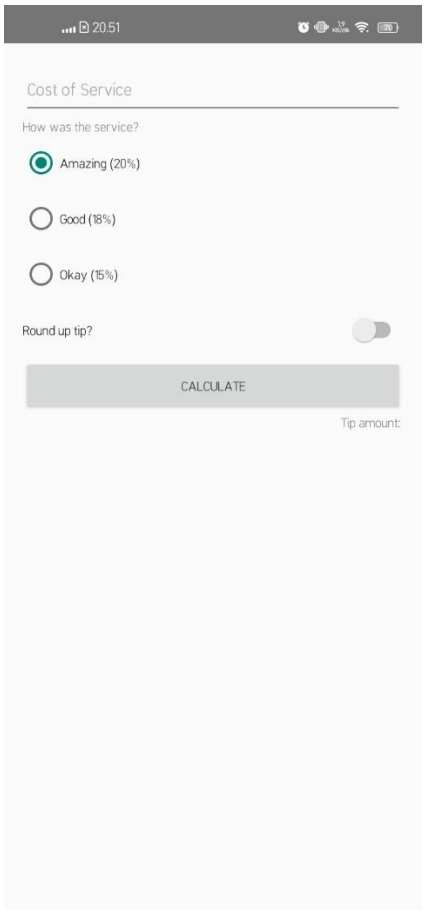
Tabel 3. Source Code Jawaban Soal 1

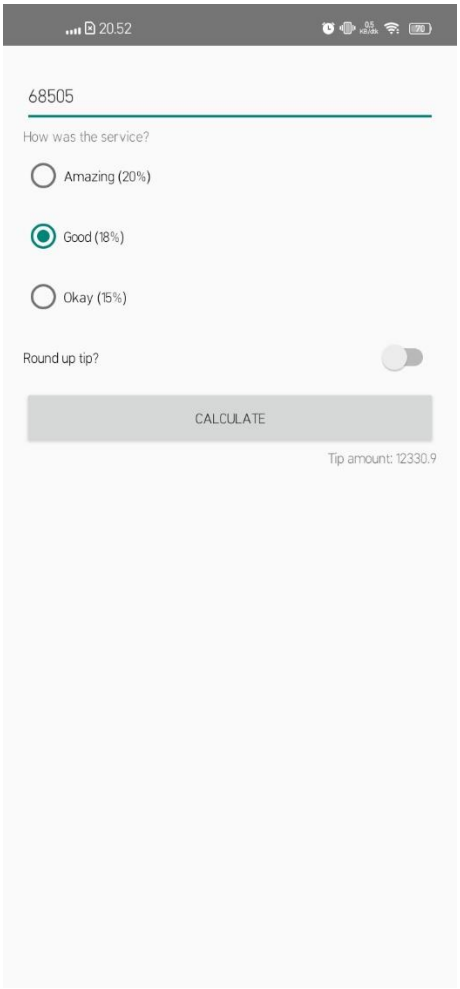
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3
```

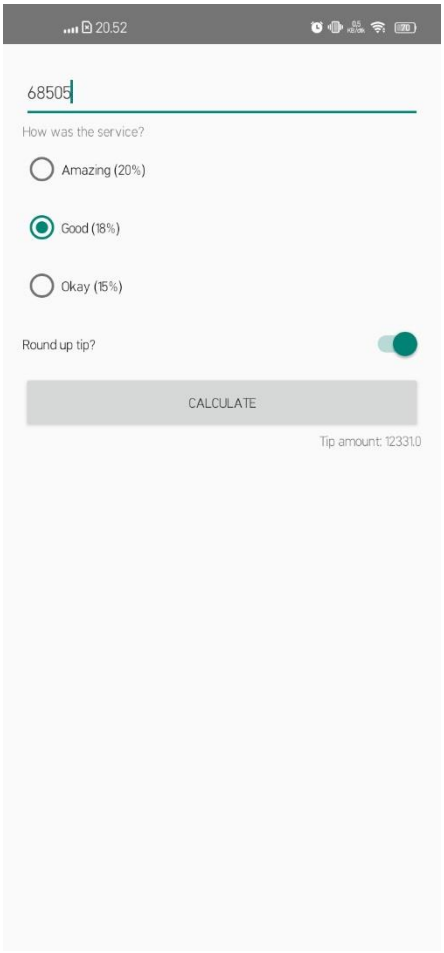
```
4   xmlns:android="http://schemas.android.com/apk/res/android"
5       xmlns:app="http://schemas.android.com/apk/res-auto"
6       xmlns:tools="http://schemas.android.com/tools"
7       android:id="@+id/main"
8       android:layout_width="match_parent"
9       android:layout_height="match_parent"
10      android:padding="16dp"
11      tools:context=".MainActivity">
12
13      <EditText
14          android:id="@+id/cost_of_service"
15          android:hint="Cost of Service"
16          android:inputType="number"
17          app:layout_constraintTop_toTopOf="parent"
18          app:layout_constraintLeft_toLeftOf="parent"
19          android:layout_width="match_parent"
20          android:layout_height="48dp"
21          />
22      <TextView
23          android:id="@+id/service_question"
24          android:layout_width="wrap_content"
25          android:layout_height="wrap_content"
26          app:layout_constraintStart_toStartOf="parent"
27  app:layout_constraintTop_toBottomOf="@id/cost_of_service"
28          android:text="How was the service?"
29          />
30      <RadioGroup
31          android:id="@+id/tip_options"
32          app:layout_constraintStart_toStartOf="parent"
33  app:layout_constraintTop_toBottomOf="@id/service_question"
34          android:checkedButton="@id/option_twenty"
35          android:layout_width="wrap_content"
36          android:layout_height="wrap_content">
37
38          <RadioButton
39              android:id="@+id/option_twenty"
40              android:layout_width="wrap_content"
41              android:layout_height="48dp"
42              android:text="Amazing (20%)"
43              app:layout_constraintTop_toTopOf="parent"
44              app:layout_constraintLeft_toLeftOf="parent"
45              />
46          <RadioButton
47              android:id="@+id/option_eighteen"
48              android:layout_width="wrap_content"
49              android:layout_height="48dp"
50              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




20.52
68505
How was the service?
Amazing (20%)
Good (18%)
Okay (15%)
Round up tip?
CALCULATE
Tip amount: 12330.9



Gambar 2. Screenshot Hasil Jawaban Soal 1 Modul 2

C. Pembahasan

1. MainActivity.kt:

Pada baris [1-14], file dimulai dengan deklarasi package serta berbagai `import` yang diperlukan untuk menjalankan komponen Android, termasuk elemen-elemen dari Jetpack Compose dan View Binding. Class `MainActivity` dideklarasikan sebagai turunan dari `ComponentActivity`, yang berfungsi sebagai entry point atau titik masuk utama dari aplikasi Android tersebut.

Pada baris [16], terdapat deklarasi variabel `binding` dengan tipe `ActivityMainBinding` dan modifier `lateinit`, yang artinya variabel ini akan diinisialisasi nanti, khususnya saat layout sudah tersedia. Variabel ini digunakan untuk mengakses view dari file layout XML yang telah di-inflate.

Pada baris [18-23], method `onCreate()` dioverride untuk menjalankan beberapa proses penting saat activity dibuat. Pertama, aplikasi mengaktifkan tampilan edge-to-edge dengan memanggil `enableEdgeToEdge()`. Selanjutnya, layout XML di-inflate ke dalam objek `binding` menggunakan `ActivityMainBinding.inflate()`, kemudian root view dari binding diatur sebagai content view aplikasi dengan `setContentView()`. Terakhir, tombol "calculate" pada layout diberikan listener (`setOnClickListener`) yang akan memicu pemanggilan fungsi `hitung()` ketika ditekan.

Pada baris [24-38], fungsi `hitung()` berisi logika utama untuk melakukan perhitungan tip. Pertama, nilai `cost` diambil dari input `EditText` dan dikonversi ke dalam tipe `Double`. Kemudian, pilihan persentase tip diambil dari `RadioGroup` dan ditentukan menggunakan ekspresi `when`, berdasarkan `id` dari `RadioButton` yang dipilih. Setelah itu, nilai tip dihitung dengan mengalikan persentase yang sesuai dengan nilai cost. Jika `Switch` bernama `roundUp` dalam keadaan aktif, maka hasil tip akan dibulatkan ke atas menggunakan fungsi `ceil()`. Jika nilai `cost` kurang dari atau sama dengan nol, maka akan ditampilkan pesan kesalahan menggunakan `Toast`. Akhirnya, hasil perhitungan tip ditampilkan pada `TextView` dengan id `total`.

2. activity_main.xml

Pada baris [1-10], layout utama aplikasi menggunakan `ConstraintLayout` sebagai root view dengan ID `main`, dan diberi padding sebesar 16dp di keempat sisinya untuk memberikan jarak yang nyaman terhadap tepi layar. Layout ini dihubungkan dengan konteks `MainActivity`, yang berarti semua komponen yang berada di dalamnya akan mengikuti logika dan siklus hidup dari activity tersebut.

Pada baris [12-20], terdapat elemen `EditText` yang digunakan untuk input nilai "cost of service". Komponen ini memiliki ID `cost_of_service`, dilengkapi dengan hint text "Cost of Service" sebagai petunjuk bagi pengguna, dan menggunakan `inputType` number agar hanya menerima input berupa angka. Ukuran tinggi ditetapkan sebesar 48dp dan lebar `match_parent` agar memanjang sesuai lebar layar.

Selanjutnya, pada baris [21-27], terdapat `TextView` dengan teks pertanyaan "How was the service?" yang diletakkan dengan constraint tepat di bawah `EditText` sebelumnya, berfungsi sebagai pengantar untuk opsi pemilihan tip.

Pada baris [28-56], terdapat `RadioGroup` yang terdiri dari tiga `RadioButton` untuk memilih persentase tip. Salah satu tombol radio, yaitu `option_twenty` (20%), disetel sebagai default terpilih. Setiap `RadioButton` menampilkan teks dan nilai persentase yang sesuai, dan seluruh grup ini di-constraint agar berada di bawah `TextView` pertanyaan.

Kemudian, pada baris [57-62], terdapat komponen `Switch` dengan teks "Round up tip?" yang memberikan opsi kepada pengguna untuk membulatkan hasil tip ke atas. Switch ini juga diposisikan dengan constraint di bawah `RadioGroup`.

Pada baris [63-68], terdapat `Button` dengan ID `calculate`, yang digunakan untuk memicu proses perhitungan tip. Tombol ini memiliki lebar `match_parent` dan dikonfigurasi agar berada tepat di bawah `Switch`.

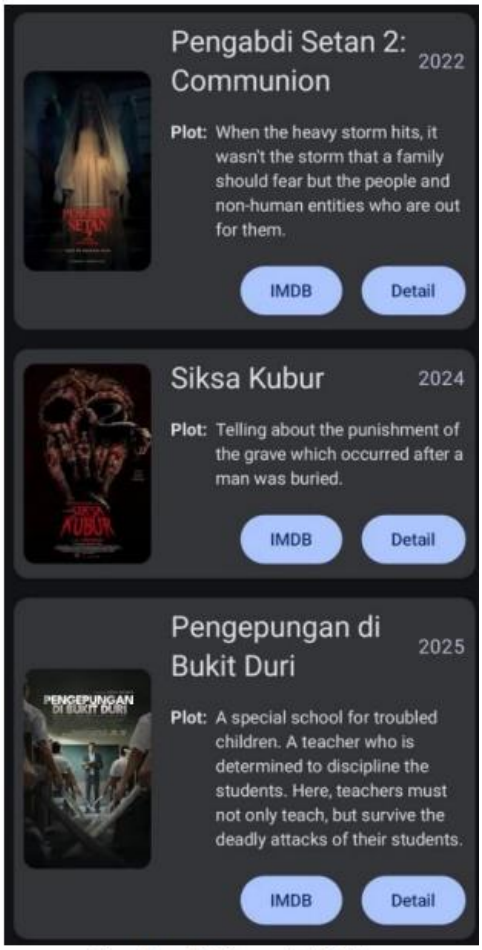
Terakhir, pada baris [69-82], terdapat dua buah `TextView` yang berfungsi untuk menampilkan hasil perhitungan. `TextView` pertama menampilkan label "Tip amount: ", dan berada di sebelah kiri, sedangkan `TextView` kedua memiliki ID `total` dan digunakan untuk menampilkan nilai hasil perhitungan tip, diletakkan di sebelah kanan. Kedua elemen ini diatur agar berada tepat di bawah tombol `calculate`, menjaga konsistensi dan kejelasan tata letak hasil output.

MODUL 3 : BUILD A SCROLLABLE LIST

SOAL 1

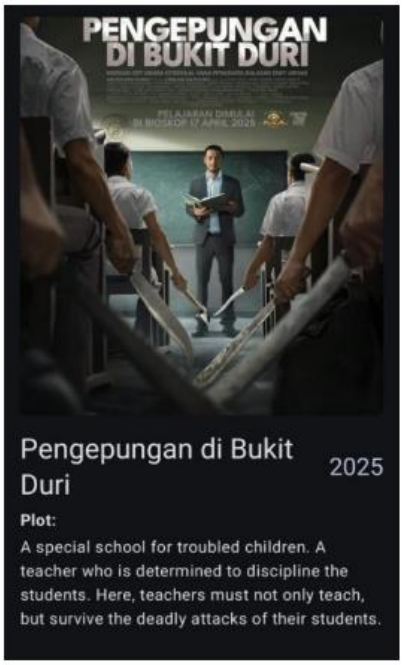
1. Buatlah sebuah aplikasi Android menggunakan XML atau Jetpack Compose yang dapat menampilkan list dengan ketentuan berikut:
    1. List menggunakan fungsi RecyclerView (XML) atau LazyColumn (Compose)
    2. List paling sedikit menampilkan 5 item. Tema item yang ingin ditampilkan bebas 3. Item pada list menampilkan teks dan gambar sesuai dengan contoh di bawah 4.Terdapat 2 button dalam list, dengan fungsi berikut:
        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

```
1   package com.allano.nongki
2
3   import android.os.Bundle
4   import androidx.appcompat.app.AppCompatActivity
5   import androidx.activity.enableEdgeToEdge
6   import com.allano.nongki.databinding.ActivityMainBinding
7
8   class MainActivity : AppCompatActivity() {
9       private lateinit var binding: ActivityMainBinding
10
11      override fun onCreate(savedInstanceState: Bundle?) {
12          super.onCreate(savedInstanceState)
13          enableEdgeToEdge()
14          binding =
    ActivityMainBinding.inflate(layoutInflater)
15          setContentView(binding.root)
16      }
17  }
```

2. HomeFragment.kt

Tabel 5. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
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.allano.nongki.databinding.FragmentHomeBinding
12
13 class HomeFragment : Fragment() {
14     private var _binding: FragmentHomeBinding? = null
15     private val binding get() = _binding!!
16
17     private val viewModel: LocationViewModel 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 =
26 LocationAdapter(viewModel.getLocations(),
   findNavController())
27         binding.recyclerView.layoutManager =
   LinearLayoutManager(requireContext())
28         binding.recyclerView.adapter = adapter
29     }
30
31     override fun onDestroyView() {
32         super.onDestroyView()
33         _binding = null
34     }
35
36 }
```

3. **LocationViewModel.kt**

Tabel 6. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.app.Application
4   import androidx.lifecycle.AndroidViewModel
5
6   class LocationViewModel(application: Application) :
    AndroidViewModel(application){
7       private val context =
    getApplication<Application>().applicationContext
8
9       fun getLocations(): List<LocationModel> {
10          val nama =
    context.resources.getStringArray(R.array.data_name)
11          val deskripsi =
    context.resources.getStringArray(R.array.data_description)
12          val rating =
    context.resources.getStringArray(R.array.data_rating)
13          val link =
    context.resources.getStringArray(R.array.data_link)
14          val fotoResId = listOf(
15              R.drawable.image1,
16              R.drawable.image2,
17              R.drawable.image3,
18              R.drawable.image4,
19              R.drawable.image5
20          )
21
22          return nama.indices.map { i ->
23              LocationModel(
24                  nama[i],
25                  fotoResId[i],
26                  deskripsi[i],
27                  rating[i],
28                  link[i]
29              )
30          }
31      }
32  }
```

4. **LocationAdapter.kt**

Tabel 7. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.content.Intent
4  import android.view.LayoutInflater
5  import android.view.ViewGroup
6  import androidx.navigation.NavController
7  import androidx.recyclerview.widget.RecyclerView
8  import com.allano.nongki.databinding.ItemLocationBinding
9  import androidx.core.net.toUri
10
11 class LocationAdapter(
12     private val items: List<LocationModel>,
13     private val navController: NavController
14 ) : RecyclerView.Adapter<LocationAdapter.LocationViewHolder>() {
15
16     class LocationViewHolder(val binding: ItemLocationBinding) :
   RecyclerView.ViewHolder(binding.root)
17
18     override fun onCreateViewHolder(parent: ViewGroup, viewType:
   Int): LocationViewHolder {
19         val binding =
   ItemLocationBinding.inflate(LayoutInflater.from(parent.context),
   parent, false)
20         return LocationViewHolder(binding)
21     }
22
23     override fun onBindViewHolder(holder: LocationViewHolder,
   position: Int) {
24         val item = items[position]
25         holder.binding.textViewName.text = item.nama
26         holder.binding.rating.text = item.rating
27
   holder.binding.imageView.setImageResource(item.fotoResId)
28
29         holder.binding.buttonBrowser.setOnClickListener {
30             val url = item.link
31             val intent = Intent(Intent.ACTION_VIEW, url.toUri())
32             it.context.startActivity(intent)
33         }
34
35         holder.binding.buttonDetail.setOnClickListener {
36             val bundle = android.os.Bundle().apply {
37                 putInt("fotoResId", item.fotoResId)
38                 putString("nama", item.nama)
39                 putString("deskripsi", item.deskripsi)
40             }
```

```
41              navController.navigate(R.id.detailFragment, bundle)
42          }
43      }

        override fun getItemCount() = items.size
44  }
45
46
```

5. DetailFragment.kt

Tabel 8. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
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 com.allano.nongki.databinding.FragmentDetailBinding
9
10  class DetailFragment: Fragment() {
11      private var _binding: FragmentDetailBinding? = null
12      private val binding get() = _binding!!
13
14      override fun onCreateView(inflater: LayoutInflater,
    container: ViewGroup?, savedInstanceState: Bundle?):
    View?{
15          _binding = FragmentDetailBinding.inflate(inflater,
    container, false)
16          return binding.root
17      }
18
19      override fun onViewCreated(view: View,
    savedInstanceState: Bundle?) {
20          val nama = arguments?.getString("nama") ?: "Tidak
    ada nama"
21          val deskripsi = arguments?.getString("deskripsi")
    ?: "Tidak ada deskripsi"
22          val fotoResId = arguments?.getInt("fotoResId") ?:
    R.drawable.image5
23         binding.imageViewDetail.setImageResource(fotoResId)
24          binding.titleViewDetail.text = nama
25          binding.textViewDetail.text = deskripsi
26      }
27
```

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

6. LocationAdapter.kt

Tabel 9. Source Code Jawaban Soal 1

```
package com.allano.nongki

data class LocationModel (
    val nama: String,
    val fotoResId: Int,
    val deskripsi: String,
    val rating: String,
    val link: String
)
```

7. Activity_main.xml

Tabel 10. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
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/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    app:navGraph="@navigation/nav_graph"
    app:defaultNavHost="true" />
```

8. Fragment_home.xml

Tabel 11. Source Code Jawaban Soal 1

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

```

```
8       <androidx.recyclerview.widget.RecyclerView
9           android:id="@+id/recyclerView"
10          android:layout_width="match_parent"
11          android:layout_height="match_parent"/>
12  </LinearLayout>
```

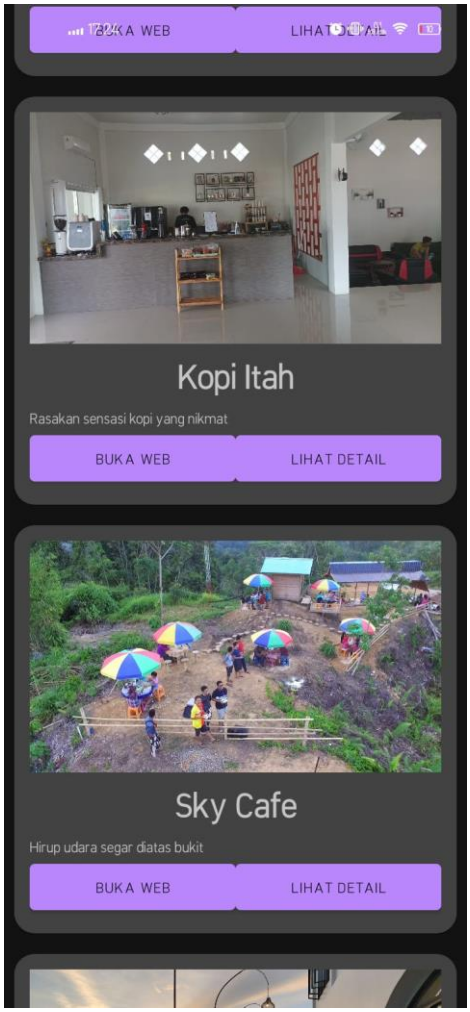
9. Fragment_detail.xml

Tabel 12. Source Code Jawaban Soal 1

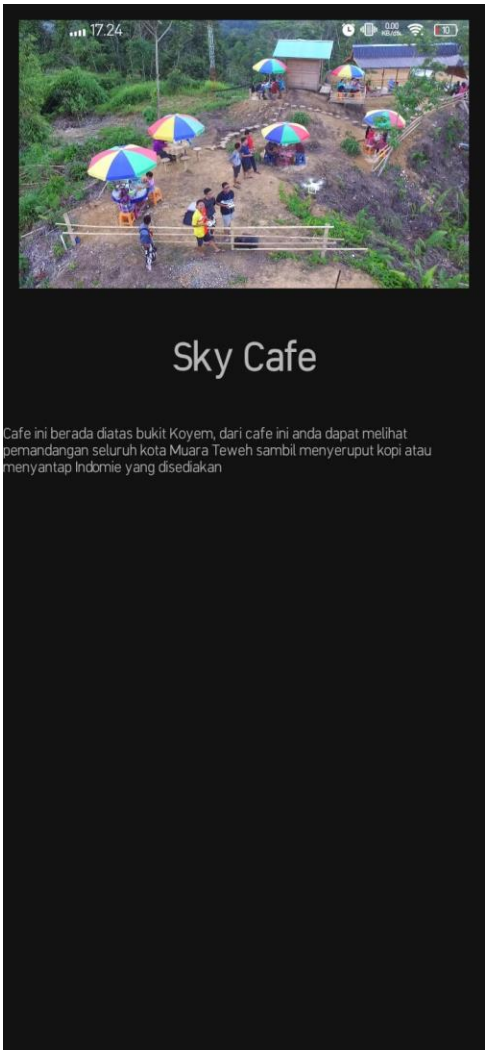
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <FrameLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       xmlns:app="http://schemas.android.com/apk/res-auto">
7       <androidx.constraintlayout.widget.ConstraintLayout
8           android:layout_width="match_parent"
9           android:layout_height="match_parent">
10      <ImageView
11          android:id="@+id/imageViewDetail"
12          android:layout_width="match_parent"
13          android:layout_height="200dp"
14          android:layout_margin="12dp"
15          android:scaleType="centerCrop"
16          android:layout_marginBottom="16dp"
17          app:layout_constraintTop_toTopOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintEnd_toEndOf="parent"/>
20      <TextView
21          android:id="@+id/titleViewDetail"
22          android:layout_width="match_parent"
23          android:layout_height="100dp"
24          android:gravity="center"
25          android:textStyle="bold"
26          android:text="detail"
27
28  app:layout_constraintTop_toBottomOf="@+id/imageViewDetail"
29          app:layout_constraintStart_toStartOf="parent"
30          app:layout_constraintEnd_toEndOf="parent"
31          android:textSize="36sp"/>
32      <TextView
33          android:id="@+id/textViewDetail"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:text="Detail"
37          android:layout_gravity="center"
```

```
38
39  app:layout_constraintTop_toBottomOf="@+id/titleViewDetail"
40          app:layout_constraintStart_toStartOf="parent"
41          app:layout_constraintEnd_toEndOf="parent"/>
42      </androidx.constraintlayout.widget.ConstraintLayout>
43  </FrameLayout>
```

B. Output Program



Gambar 3. Screenshot Hasil Jawaban Soal 1



Gambar 4. Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt:

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace dari aplikasi. Selanjutnya, pada baris [2] hingga [4], berbagai library penting diimpor, termasuk komponen Android dan view binding yang diperlukan untuk mengelola tampilan. Pada baris [5], dideklarasikan kelas `MainActivity` yang merupakan turunan dari `AppCompatActivity`, yang berfungsi sebagai aktivitas utama aplikasi. Pada baris [6], variabel `binding` dideklarasikan dengan tipe `ActivityMainBinding`, yang memungkinkan akses langsung ke elemen-elemen layout XML melalui view binding.

Pada baris [8-11], metode `onCreate()` diimplementasikan sebagai titik awal saat aktivitas dijalankan. Pada baris [9], `super.onCreate(savedInstanceState)` dipanggil untuk menjalankan inisialisasi dasar dari superclass. Di baris [10], `enableEdgeToEdge()` digunakan untuk mengaktifkan tampilan layar penuh (edge-to-edge display), memberikan pengalaman visual yang lebih modern. Kemudian, pada baris [11], layout di-inflate menggunakan `ActivityMainBinding.inflate(layoutInflater)` agar binding dapat digunakan. Akhirnya, pada baris [12], `setContentView(binding.root)` dipanggil untuk menampilkan root view dari layout sebagai tampilan utama aktivitas.

2. HomeFragment.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace dari fragment ini. Pada baris [2] hingga [8], sejumlah library penting diimpor untuk mendukung fungsionalitas fragment, seperti pengelolaan view, navigasi, dan RecyclerView. Pada baris [9], kelas `HomeFragment` dideklarasikan sebagai turunan dari `Fragment`, yang merepresentasikan salah satu tampilan (UI screen) dalam aplikasi.

Pada baris [10] dan [11], variabel `binding` dideklarasikan menggunakan pola nullable dengan properti `_binding`, serta getter non-null `binding` untuk akses yang aman. Pada baris [12], `viewModel` diinisialisasi menggunakan `by viewModels()` yang otomatis mengelola lifecycle sesuai fragment.

Pada baris [14–18], metode `onCreateView()` diimplementasikan untuk mengatur tampilan awal fragment. Di baris [16], layout di-inflate menggunakan `FragmentHomeBinding.inflate()`, dan hasil binding disimpan ke `_binding`. Kemudian, pada baris [17], root view dari binding dikembalikan agar fragment dapat menampilkan UI-nya.

Pada baris [20–23], metode `onViewCreated()` digunakan untuk inisialisasi lanjutan setelah view terbentuk. Pada baris [21], `LocationAdapter` diinisialisasi dengan data lokasi dari `viewModel` dan `NavController` untuk mendukung navigasi. Baris [22] mengatur `LinearLayoutManager` pada `RecyclerView` agar menampilkan item secara vertikal, dan baris [23] menetapkan adapter ke `RecyclerView`.

Terakhir, pada baris [25–27], metode `onDestroyView()` bertugas membersihkan binding dengan menyetel `_binding` menjadi `null` agar tidak terjadi memory leak setelah view dihancurkan.

3. LocationViewModel.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai penanda namespace untuk kelas ini. Pada baris [2] dan [3], diimpor library yang dibutuhkan, termasuk `android.app.Application` dan `androidx.lifecycle.AndroidViewModel`, yang diperlukan untuk membuat ViewModel berbasis konteks aplikasi. Pada baris [4], kelas `LocationViewModel` dideklarasikan sebagai turunan dari `AndroidViewModel`, memungkinkan ViewModel mengakses `Application` context secara langsung.

Pada baris [5], konteks aplikasi diambil dari parameter konstruktor dan disimpan dalam variabel `context` untuk digunakan dalam pengambilan resource. Pada baris [7], fungsi `getLocations()` dideklarasikan, yang mengembalikan daftar `LocationModel`.

Pada baris [8–11], data berupa nama tempat, deskripsi, dan lokasi diambil dari resource `string-array` menggunakan `context.resources.getStringArray()`. Selanjutnya, pada baris [12–17], daftar gambar tempat disusun dalam bentuk list resource ID dari drawable.

Pada baris [19–25], dilakukan pemetaan data ke dalam objek `LocationModel` dengan menggunakan indeks array. Setiap elemen dalam list dibentuk dari gabungan elemen nama, deskripsi, lokasi, dan gambar berdasarkan posisi yang sama di masing-masing array, lalu dikembalikan sebagai list `LocationModel`.

4. LocationAdapter.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace utama untuk file ini. Pada baris [2–6], berbagai library penting diimpor, termasuk komponen RecyclerView, tampilan View, binding untuk layout, navigasi, dan Intent untuk membuka browser.

Pada baris [7], kelas `LocationAdapter` dideklarasikan sebagai turunan dari `RecyclerView.Adapter`, yang bertugas menampilkan daftar lokasi menggunakan ViewHolder.

Pada baris [9], inner class `LocationViewHolder` didefinisikan untuk memegang referensi view setiap item daftar, melalui view binding.

Pada baris [14–16], metode `onCreateViewHolder` diimplementasikan untuk meng-inflate layout XML dari item dan membungkusnya dalam `LocationViewHolder`.

Pada baris [18–35], metode `onBindViewHolder` digunakan untuk mengisi data dan menangani interaksi pengguna pada setiap item dalam daftar.

Pada baris [20–22], data dari objek `LocationModel` seperti nama, deskripsi, dan gambar dimasukkan ke dalam view-item masing-masing.

Pada baris [24–27], ketika tombol "browser" ditekan, akan dibuat `Intent` dengan `ACTION_VIEW` untuk membuka link lokasi di browser.

Pada baris [29–34], ketika tombol "detail" ditekan, `NavController` digunakan untuk navigasi ke fragment detail, membawa objek `LocationModel` sebagai argumen.

Pada baris [37], metode `getItemCount` mengembalikan jumlah total data dalam list, yang digunakan oleh RecyclerView untuk menentukan jumlah item yang perlu ditampilkan.

5. DetailFragment.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace untuk kelas ini. Pada baris [2–5], library-library yang diperlukan diimpor, termasuk komponen Fragment, View, dan binding.

Pada baris [6], kelas `DetailFragment` dideklarasikan sebagai turunan dari `Fragment`, yang berfungsi untuk menampilkan detail lokasi.

Pada baris [7–8], variabel binding didefinisikan untuk mengakses view dari layout dengan pola nullable dan non-null getter (`_binding` dan `binding`).

Pada baris [10–14], method `onCreateView` diimplementasikan untuk meng-inflate layout menggunakan `FragmentDetailBinding`, kemudian mengembalikan root view untuk ditampilkan.

Pada baris [16–23], method `onViewCreated` digunakan untuk mengambil objek `LocationModel` dari arguments (menggunakan `arguments?.getParcelable`), lalu data seperti nama, deskripsi, gambar, dan lokasi di-set ke komponen view yang sesuai.

Pada baris [25–27], method `onDestroyView` membersihkan referensi `_binding` untuk menghindari memory leak saat fragment dihancurkan.

6. LocationModel.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace untuk file ini.

Pada baris [3], `data class LocationModel` didefinisikan sebagai kelas data yang digunakan untuk merepresentasikan informasi lokasi nongki.

Pada baris [4–8], constructor utama dari `LocationModel` dideklarasikan dengan lima parameter:

- `nama` bertipe `String`, menyimpan nama tempat
- `fotoResId` bertipe `Int`, menyimpan ID resource gambar
- `deskripsi` bertipe `String`, menyimpan deskripsi tempat
- `rating` bertipe `String`, menyimpan nilai rating tempat
- `link` bertipe `String`, menyimpan URL lokasi atau informasi tambahan

7. Activity_main.xml

Pada baris [1], deklarasi versi XML dan encoding (`<?xml version="1.0" encoding="utf-8"?>`) ditulis untuk memastikan file diproses sebagai dokumen XML dengan encoding UTF-8.

Pada baris [2], elemen root berupa `androidx.fragment.app.FragmentContainerView` digunakan sebagai kontainer untuk menampilkan fragment secara dinamis.

Pada baris [3–5], namespace XML seperti `xmlns:android`, `xmlns:app`, dan `xmlns:tools` diimpor untuk mendukung atribut khusus Android dan AndroidX.

Pada baris [6], atribut `android:id` di-set ke `@+id/nav_host_fragment` untuk mengidentifikasi kontainer fragment ini dalam kode.

Pada baris [7], atribut `android:name` di-set ke `androidx.navigation.fragment.NavHostFragment` yang berfungsi sebagai host untuk fragment navigasi.

Pada baris [8–9], `android:layout_width` dan `android:layout_height` diatur ke `match_parent` agar kontainer memenuhi seluruh layar.

Pada baris [10], `tools:context` menunjuk ke `MainActivity`, digunakan untuk pratinjau layout di Android Studio.

Pada baris [11], `app:navGraph` di-set ke `@navigation/nav_graph`, yaitu referensi ke file

navigasi XML yang mendefinisikan alur navigasi aplikasi. Pada baris [12], `app:defaultNavHost` di-set ke `true` agar `NavHostFragment` ini menjadi host utama untuk menangani aksi navigasi seperti tombol kembali.

8. Fragment_home.xml

Pada baris [1], terdapat deklarasi versi XML dan encoding (`<?xml version="1.0" encoding="utf-8"?>`) untuk memastikan file dibaca dengan benar sebagai dokumen XML berstandar UTF-8.
Pada baris [2], elemen root menggunakan `LinearLayout` yang berfungsi sebagai wadah vertikal atau horizontal bagi komponen UI di dalamnya.
Pada baris [3] dan [4], `android:orientation` di-set ke `vertical` untuk menata elemen secara vertikal, dan `layout_width` serta `layout_height` di-set ke `match_parent` agar memenuhi layar sepenuhnya.
Pada baris [6], elemen `androidx.recyclerview.widget.RecyclerView` digunakan untuk menampilkan daftar item secara efisien dalam jumlah besar.
Pada baris [7], `android:id` di-set ke `@+id/recyclerView` agar dapat diakses melalui kode Kotlin atau Java.
Pada baris [8] dan [9], `layout_width` dan `layout_height` untuk `RecyclerView` juga di-set ke `match_parent`, sehingga komponen ini mengisi seluruh ruang yang tersedia dalam `LinearLayout`.

9. Fragment_detail.xml

Pada baris [1], terdapat deklarasi XML `<?xml version="1.0" encoding="utf-8"?>` yang merupakan standar awal untuk mendefinisikan dokumen XML dengan encoding UTF-8. Pada baris [2], elemen root dari layout ini adalah `FrameLayout`, sebuah container yang memungkinkan penumpukan elemen-elemen anak secara fleksibel. Pada baris [3] hingga [5], layout disiapkan dengan atribut `layout_width` dan `layout_height` yang di-set ke `match_parent`, serta definisi namespace Android (`xmlns:android`) dan `app` untuk mendukung ConstraintLayout.
Pada baris [7], `ConstraintLayout` digunakan sebagai child view utama dalam `FrameLayout`, karena layout ini mendukung penempatan elemen-elemen UI yang

responsif berdasarkan constraint antar elemen. Layout ini digunakan untuk menyusun komponen UI seperti gambar, judul, dan deskripsi secara terstruktur.

Pada baris [9], terdapat `ImageView` yang digunakan untuk menampilkan gambar lokasi secara detail. Komponen ini diberi ID `@+id/imageViewDetail`, dengan tinggi layout `200dp`, margin `12dp` di sekelilingnya, dan properti `scaleType` diset ke `centerCrop` agar gambar dapat mengisi ruang yang tersedia tanpa distorsi. Posisi `ImageView` ini diletakkan di bagian atas layout dan diatur agar menempel ke sisi atas dan horizontal parent dengan menggunakan constraint.

Pada baris [18], `TextView` digunakan untuk menampilkan judul dari lokasi yang dipilih. Teks judul ditampilkan dengan gaya huruf tebal (`textStyle="bold"`), ukuran huruf besar (`textSize="36sp"`), dan rata tengah (`gravity="center"`) agar terlihat menonjol dan menarik perhatian pengguna. `TextView` ini diposisikan tepat di bawah `ImageView` menggunakan `layout_constraintTop_toBottomOf`.

Pada baris [27], terdapat `TextView` tambahan yang digunakan untuk menampilkan deskripsi dari lokasi secara lebih rinci. Komponen ini memiliki `layout_width wrap_content` agar lebarnya menyesuaikan dengan panjang teks, dan ditempatkan di bawah judul menggunakan constraint `layout_constraintTop_toBottomOf`. Dengan susunan ini, pengguna dapat melihat gambar, nama tempat, dan informasi deskriptif secara berurutan dan rapi di tampilan detail aplikasi

SOAL 2

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

A. Pembahasan

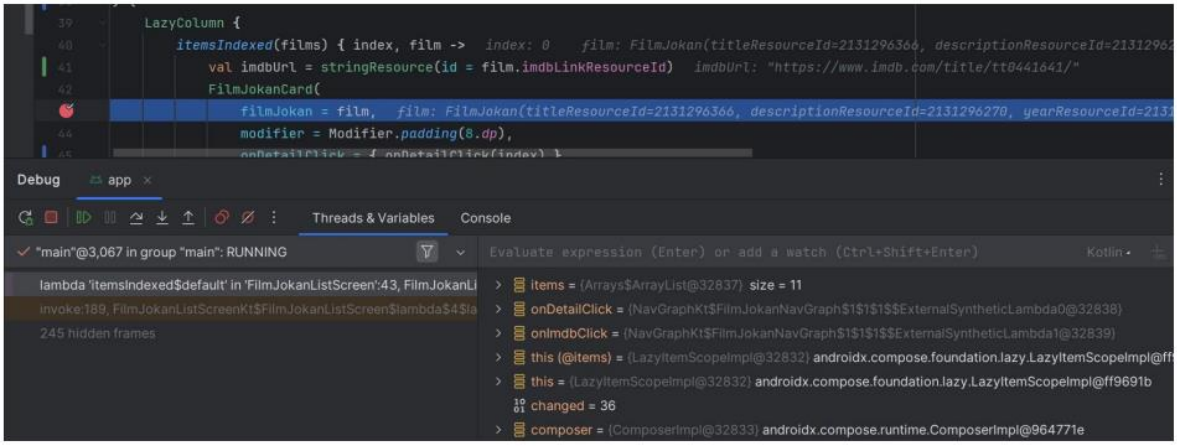
RecyclerView memiliki performa yang tinggi, khususnya untuk menangani dataset yang besar. Meskipun bersifat boiler-plate yang panjang, pengembang memiliki kendali penuh atas UI. RecyclerView juga disupport dan digunakan secara luas oleh komunitas android.

MODUL 4 : VIEWMODEL AND DEBUGGING

SOAL 1

1. Lanjutkan aplikasi Android berbasis XML dan Jetpack Compose yang sudah dibuat pada Modul 3 dengan menambahkan modifikasi sesuai ketentuan berikut:
    a. Buatlah sebuah ViewModel untuk menyimpan dan mengelola data dari list item. Data tidak boleh disimpan langsung di dalam Fragment atau Activity.
    b. Gunakan ViewModelFactory dalam pembuatan ViewModel
    c. Gunakan StateFlow untuk mengelola event onClick dan data list item dari ViewModel ke Fragment
    d. gunakan logging untuk event berikut:
        a. Log saat data item masuk ke dalam list
        b. Log saat tombol Detail dan tombol Explicit Intent ditekan
        c. Log data dari list yang dipilih ketika berpindah ke halaman Detail
    e. Gunakan tool Debugger di Android Studio untuk melakukan debugging pada aplikasi. Cari setidaknya satu breakpoint yang relevan dengan aplikasi. Lalu, gunakan fitur Step Into, Step Over, dan Step Out. Setelah itu, jelaskan fungsi Debugger, cara menggunakan Debugger, serta fitur Step Into, Step Over, dan Step Out
2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

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



A. Source Code

1. MainActivity.kt

Tabel 13. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.os.Bundle
4  import androidx.appcompat.app.AppCompatActivity
5  import androidx.activity.enableEdgeToEdge
6  import com.allano.nongki.databinding.ActivityMainBinding
7
8  class MainActivity : AppCompatActivity() {
9      private lateinit var binding: ActivityMainBinding
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         enableEdgeToEdge()
14         binding =
   ActivityMainBinding.inflate(layoutInflater)
15         setContentView(binding.root)
16     }
17 }
```

2. HomeFragment.kt

Tabel 14. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.os.Bundle
4  import android.util.Log
```

```
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import androidx.fragment.app.Fragment
9  import androidx.fragment.app.viewModels
10 import androidx.lifecycle.lifecycleScope
11 import androidx.navigation.fragment.findNavController
12 import androidx.recyclerview.widget.LinearLayoutManager
13 import com.allano.nongki.databinding.FragmentHomeBinding
14 import kotlinx.coroutines.flow.collect
15 import kotlinx.coroutines.launch
16
17 class HomeFragment : Fragment() {
18     private var _binding: FragmentHomeBinding? = null
19     private val binding get() = _binding!!
20     private lateinit var adapter: LocationAdapter
21
22     private val viewModel: LocationViewModel by viewModels
23 {
24
25 LocationViewModel.Factory(requireActivity().application)
26     }
27
28     override fun onCreateView(inflater: LayoutInflater,
   container: ViewGroup?, savedInstanceState: Bundle?): View?
29 {
30         _binding = FragmentHomeBinding.inflate(inflater,
   container, false)
31         return binding.root
32     }
33
34     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
35         super.onViewCreated(view, savedInstanceState)
36
37         adapter = LocationAdapter(emptyList(),
   findNavController(), viewModel)
38         binding.recyclerView.layoutManager =
   LinearLayoutManager(requireContext())
39         binding.recyclerView.adapter = adapter
40
41         lifecycleScope.launch {
42             viewModel.locations.collect { locations ->
43                 adapter.updateData(locations)
44                 Log.d("HomeFragment", "Data list
   diperbarui di UI")
45             }
```

```
46          }
47
48          lifecycleScope.launch {
49              viewModel.selectedLocation.collect { location
    ->
50                  location?.let {
51                      navigateToDetail(it)
52                  }
53              }
54          }
55      }
56
57      private fun navigateToDetail(location: LocationModel)
58  {
59          val bundle = Bundle().apply {
60              putInt("fotoResId", location.fotoResId)
61              putString("nama", location.nama)
62              putString("deskripsi", location.deskripsi)
63          }
64          Log.d("HomeFragment", "Navigasi ke detail dengan
    data: ${location.nama}")
65          findNavController().navigate(R.id.detailFragment,
    bundle)
66      }
67
68      override fun onDestroyView() {
69          super.onDestroyView()
70          _binding = null
71      }
72  }
```

3. LocationViewModel.kt

Tabel 15. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.app.Application
4   import android.util.Log
5   import androidx.lifecycle.AndroidViewModel
6   import androidx.lifecycle.ViewModel
7   import androidx.lifecycle.ViewModelProvider
8   import androidx.lifecycle.viewModelScope
9   import kotlinx.coroutines.flow.MutableStateFlow
10  import kotlinx.coroutines.flow.StateFlow
11  import kotlinx.coroutines.flow.asStateFlow
```

```
12 import kotlinx.coroutines.launch
13
14 class LocationViewModel(application: Application) :
   AndroidViewModel(application) {
15     private val context =
   getApplication<Application>().applicationContext
16     private val _locations =
   MutableStateFlow<List<LocationModel>>(emptyList())
17     val locations: StateFlow<List<LocationModel>> =
   _locations.asStateFlow()
18
19     private val _selectedLocation =
   MutableStateFlow<LocationModel?>(null)
20     val selectedLocation: StateFlow<LocationModel?> =
   _selectedLocation.asStateFlow()
21
22     init {
23         loadLocations()
24     }
25
26     private fun loadLocations() {
27         viewModelScope.launch {
28             val nama =
   context.resources.getStringArray(R.array.data_name)
29             val deskripsi =
   context.resources.getStringArray(R.array.data_description)
30             val rating =
   context.resources.getStringArray(R.array.data_rating)
31             val link =
   context.resources.getStringArray(R.array.data_link)
32             val fotoResId = listOf(
33                 R.drawable.image1,
34                 R.drawable.image2,
35                 R.drawable.image3,
36                 R.drawable.image4,
37                 R.drawable.image5
38             )
39
40             val locationsList = nama.indices.map { i ->
41                 LocationModel(
42                     nama[i],
43                     fotoResId[i],
44                     deskripsi[i],
45                     rating[i],
46                     link[i]
47                 ).also {
48                     Log.d("LocationViewModel", "Data item
```

```
    ditambahkan: ${it.nama}")
49                  }
50              }
51
52              _locations.value = locationsList
53              Log.d("LocationViewModel", "Data list selesai
    dimuat, total: ${locationsList.size} item")
54          }
55      }
56
57      fun selectLocation(location: LocationModel) {
58          _selectedLocation.value = location
59          Log.d("LocationViewModel", "Item dipilih:
    ${location.nama}")
60      }
61
62      class Factory(private val application: Application) :
    ViewModelProvider.Factory {
63          @Suppress("UNCHECKED_CAST")
64          override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
65              if
    (modelClass.isAssignableFrom(LocationViewModel::class.java))
66  {
67                  return LocationViewModel(application) as T
68              }
69              throw IllegalArgumentException("Unknown
    ViewModel class")
70          }
71      }
72  }
```

4. LocationAdapter.kt

Tabel 16. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   import android.content.Intent
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.ViewGroup
7   import androidx.navigation.NavController
8   import androidx.recyclerview.widget.RecyclerView
9   import com.allano.nongki.databinding.ItemLocationBinding
10  import androidx.core.net.toUri
11
```

```
12 class LocationAdapter(
13     private var items: List<LocationModel>,
14     private val navController: NavController,
15     private val viewModel: LocationViewModel
16 ) : RecyclerView.Adapter<LocationAdapter.LocationViewHolder>() {
17
18     class LocationViewHolder(val binding: ItemLocationBinding) :
   RecyclerView.ViewHolder(binding.root)
19
20     fun updateData(newItems: List<LocationModel>) {
21         items = newItems
22         notifyDataSetChanged()
23     }
24
25     override fun onCreateViewHolder(parent: ViewGroup, viewType:
   Int): LocationViewHolder {
26         val binding =
   ItemLocationBinding.inflate(LayoutInflater.from(parent.context),
   parent, false)
27         return LocationViewHolder(binding)
28     }
29
30     override fun onBindViewHolder(holder: LocationViewHolder,
   position: Int) {
31         val item = items[position]
32         holder.binding.textViewName.text = item.nama
33         holder.binding.rating.text = item.rating
34
   holder.binding.imageView.setImageResource(item.fotoResId)
35
36         holder.binding.buttonBrowser.setOnClickListener {
37             val url = item.link
38             Log.d("LocationAdapter", "Tombol browser ditekan
   untuk: ${item.nama}, URL: $url")
39             val intent = Intent(Intent.ACTION_VIEW, url.toUri())
40             it.context.startActivity(intent)
41         }
42
43         holder.binding.buttonDetail.setOnClickListener {
44             Log.d("LocationAdapter", "Tombol detail ditekan
   untuk: ${item.nama}")
45             viewModel.selectLocation(item)
46         }
47     }
48
49     override fun getItemCount() = items.size
50 }
```

5. DetailFragment.kt

Tabel 17. Source Code Jawaban Soal 1

```
1  package com.allano.nongki
2
3  import android.os.Bundle
4  import android.util.Log
5  import android.view.LayoutInflater
6  import android.view.View
7  import android.view.ViewGroup
8  import androidx.fragment.app.Fragment
9  import com.allano.nongki.databinding.FragmentDetailBinding
10
11 class DetailFragment: Fragment() {
12     private var _binding: FragmentDetailBinding? = null
13     private val binding get() = _binding!!
14
15     override fun onCreateView(inflater: LayoutInflater,
   container: ViewGroup?, savedInstanceState: Bundle?): View?
16 {
17         _binding = FragmentDetailBinding.inflate(inflater,
   container, false)
18         return binding.root
19     }
20
21     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
22         super.onViewCreated(view, savedInstanceState)
23
24         val nama = arguments?.getString("nama") ?: "Tidak
   ada nama"
25         val deskripsi = arguments?.getString("deskripsi")
   ?: "Tidak ada deskripsi"
26         val fotoResId = arguments?.getInt("fotoResId") ?:
   R.drawable.image5
27
28         Log.d("DetailFragment", "Menampilkan detail untuk:
   $nama")
29         Log.d("DetailFragment", "Deskripsi: $deskripsi")
30         Log.d("DetailFragment", "Foto Resource ID:
   $fotoResId")
31
32
33 binding.imageViewDetail.setImageResource(fotoResId)
34         binding.titleViewDetail.text = nama
```

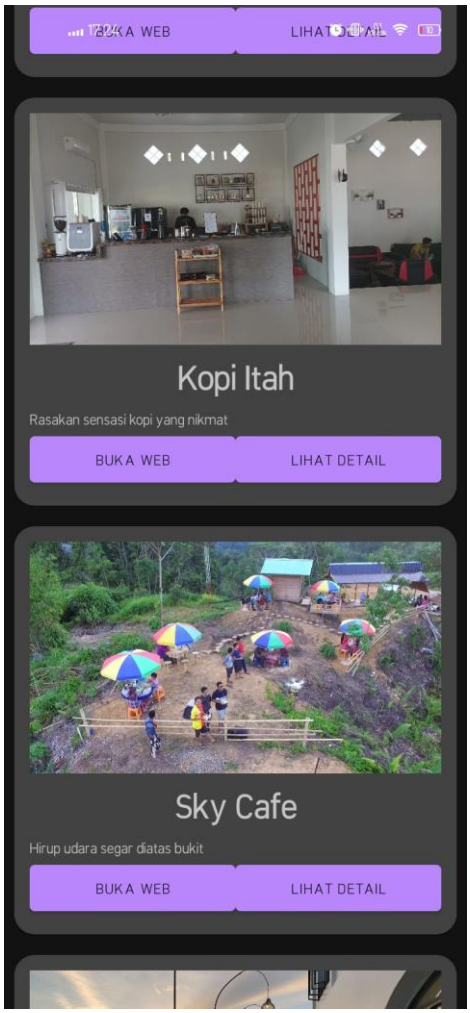
```
35          binding.textViewDetail.text = deskripsi
36      }
37
38      override fun onDestroyView() {
39          super.onDestroyView()
40          _binding = null
41      }
42  }
```

6. LocationModel.kt

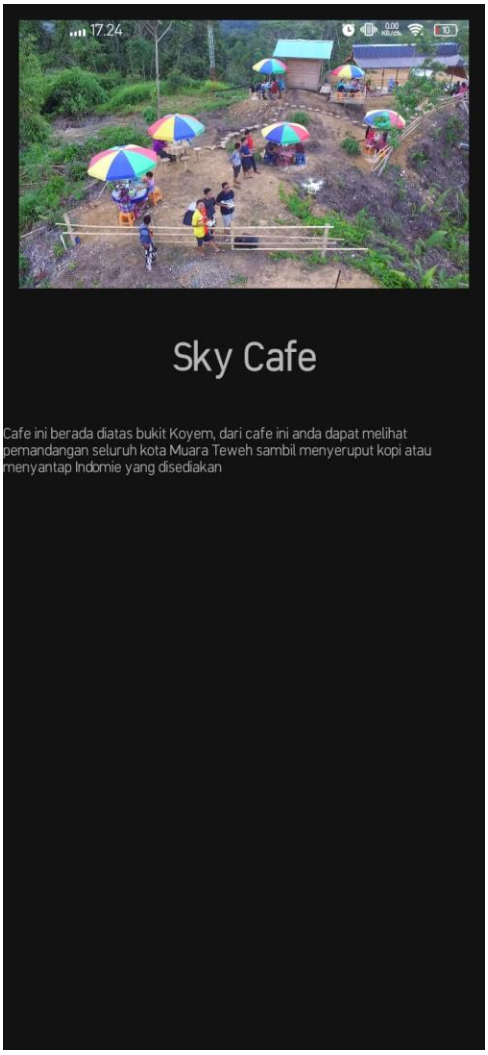
Tabel 18. Source Code Jawaban Soal 1

```
1   package com.allano.nongki
2
3   data class LocationModel (
4       val nama: String,
5       val fotoResId: Int,
6       val deskripsi: String,
7       val rating: String,
8       val link: String
9   )
```

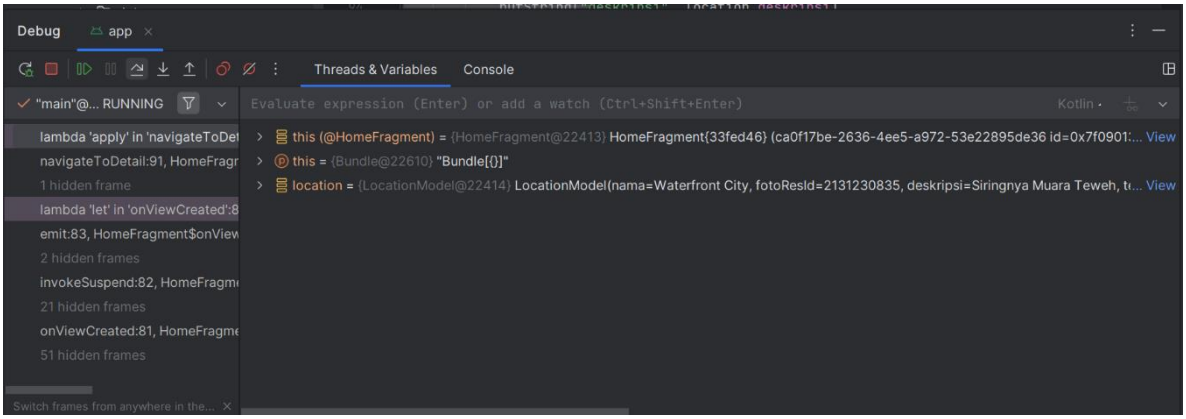
B. Output Program



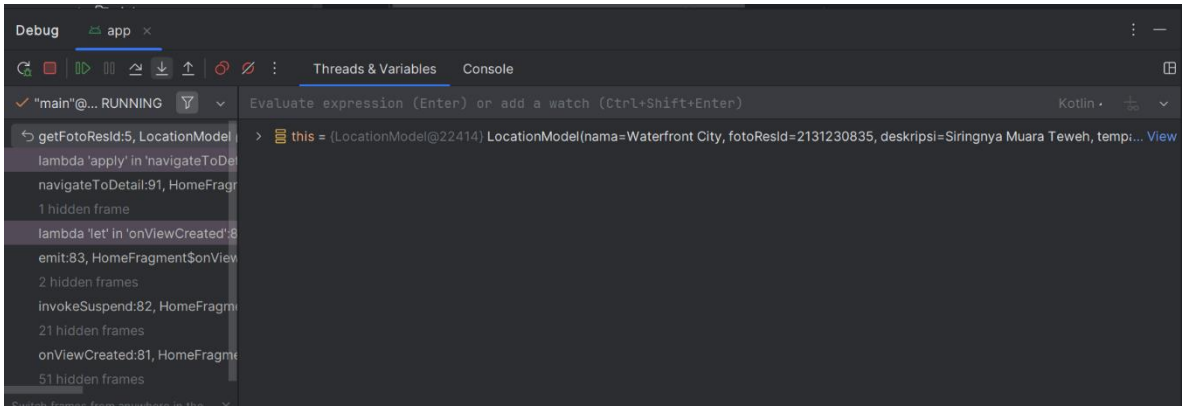
Gambar 5. Screenshot Hasil Jawaban Soal 1



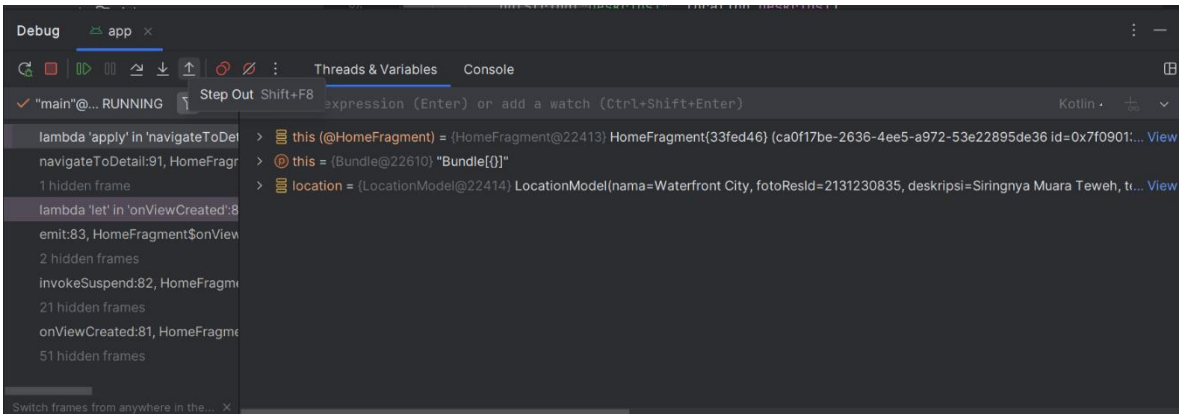
Gambar 6. Screenshot Hasil Jawaban Soal 1



Gambar 7. Debugging dengan Fitur Step Into



Gambar 8. Debugging dengan Fitur Step Over



Gambar 9. Debugging dengan Fitur Step Out

C. Pembahasan

1. MainActivity.kt:

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace dari aplikasi. Selanjutnya, pada baris [2] hingga [4], berbagai library penting diimpor, termasuk komponen Android dan view binding yang diperlukan untuk mengelola tampilan. Pada baris [5], dideklarasikan kelas `MainActivity` yang merupakan turunan dari `AppCompatActivity`, yang berfungsi sebagai aktivitas utama aplikasi. Pada baris [6], variabel `binding` dideklarasikan dengan tipe `ActivityMainBinding`, yang memungkinkan akses langsung ke elemen-elemen layout XML melalui view binding.

Pada baris [8-11], metode `onCreate()` diimplementasikan sebagai titik awal saat aktivitas dijalankan. Pada baris [9], `super.onCreate(savedInstanceState)` dipanggil untuk menjalankan inisialisasi dasar dari superclass. Di baris [10], `enableEdgeToEdge()` digunakan untuk mengaktifkan tampilan layar penuh (edge-to-

edge display), memberikan pengalaman visual yang lebih modern. Kemudian, pada baris [11], layout di-inflate menggunakan `ActivityMainBinding.inflate(layoutInflater)` agar binding dapat digunakan. Akhirnya, pada baris [12], `setContentView(binding.root)` dipanggil untuk menampilkan root view dari layout sebagai tampilan utama aktivitas.

2. HomeFragment.kt

Baris [1] mendeklarasikan `package`, dan baris [3]–[12] mengimpor library yang diperlukan seperti `ViewModel`, `LiveData`, dan `RecyclerView`. Class `HomeFragment` dideklarasikan pada baris [14] sebagai turunan dari `Fragment`.

Pada baris [15]–[18] dideklarasikan properti untuk `view binding` dan `adapter`, sedangkan `ViewModel` diinisialisasi dengan delegate `by viewModels()` (baris [20]–[22]).

Method `onCreateView()` (baris [24]–[27]) digunakan untuk *inflate* layout fragment. Method `onViewCreated()` (baris [29]–[47]) menyiapkan `RecyclerView` dan mengamati data dari `ViewModel` untuk ditampilkan secara dinamis. Method `navigateToDetail()` (baris [49]–[57]) menangani navigasi ke fragment detail dengan membawa data item terpilih melalui `Bundle`. Terakhir, method `onDestroyView()` (baris [59]–[62]) digunakan untuk membersihkan objek binding agar tidak terjadi memory leak.

3. LocationViewModel.kt

Pada baris [1] dideklarasikan `package com.allano.nongki`, sedangkan baris [3]–[11] mengimpor class yang dibutuhkan, seperti `AndroidViewModel`, `StateFlow`, dan komponen coroutine.

Class `LocationViewModel` didefinisikan pada baris [13] sebagai turunan dari `AndroidViewModel`. Baris [14]–[15] menyimpan context aplikasi. Pada baris [16]–[19], dibuat dua `StateFlow`: satu untuk daftar lokasi dan satu lagi untuk lokasi yang dipilih.

Method `loadLocations()` (baris [25]–[47]) mengambil data lokasi dari `string-array` di resources, lalu mengubahnya menjadi list `LocationModel`. Method `selectLocation()` (baris [49]–[52]) digunakan untuk mengubah nilai lokasi yang dipilih. Terakhir, inner class

`Factory` (baris [54]–[63]) digunakan untuk membuat instance `ViewModel` dengan context aplikasi.

4. LocationAdapter.kt

Baris [1] mendeklarasikan package, dan baris [3]–[9] mengimpor library yang diperlukan seperti RecyclerView dan ViewBinding. Class `LocationAdapter` didefinisikan pada baris [11] sebagai turunan dari `RecyclerView.Adapter`, yang digunakan untuk menampilkan daftar lokasi.

ViewHolder didefinisikan pada baris [18]–[20] untuk mengelola binding item layout. Method `updateData()` (baris [22]–[25]) berfungsi memperbarui data dan me-refresh tampilan. `onCreateViewHolder()` (baris [27]–[30]) membuat instance ViewHolder. `onBindViewHolder()` (baris [32]–[47]) mengikat data ke tampilan dan menangani klik. Method `getItemCount()` (baris [49]) mengembalikan jumlah item dalam list.

5. DetailFragment.kt

Baris [1] mendeklarasikan package, dan baris [3]–[7] mengimpor library yang diperlukan seperti Fragment, Bundle, dan ViewBinding. Class `DetailFragment` dideklarasikan pada baris [9] sebagai turunan dari `Fragment`. View binding dideklarasikan di baris [10]–[11] untuk mengakses elemen layout dengan aman.

Method `onCreateView()` (baris [13]–[16]) meng-inflate layout fragment. Pada `onViewCreated()` (baris [18]–[30]), data dari arguments diambil dan ditampilkan ke tampilan, seperti nama dan deskripsi lokasi. Logging digunakan untuk memastikan data diterima dengan benar. Method `onDestroyView()` (baris [32]–[35]) membersihkan binding untuk mencegah memory leak.

6. LocationModel.kt

Pada baris [1], package `com.allano.nongki` dideklarasikan sebagai namespace untuk file ini.

Pada baris [3], `data class LocationModel` didefinisikan sebagai kelas data yang digunakan untuk merepresentasikan informasi lokasi nongki.

Pada baris [4–8], constructor utama dari `LocationModel` dideklarasikan dengan lima parameter:

- `nama` bertipe `String`, menyimpan nama tempat
- `fotoResId` bertipe `Int`, menyimpan ID resource gambar
- `deskripsi` bertipe `String`, menyimpan deskripsi tempat
- `rating` bertipe `String`, menyimpan nilai rating tempat
- `link` bertipe `String`, menyimpan URL lokasi atau informasi tambahan

7. Debugger

Debugger berfungsi untuk memeriksa dan melacak jalannya program dengan cara menghentikan eksekusi aplikasi pada titik tertentu menggunakan breakpoint, melihat isi variabel, dan memantau kondisi aplikasi secara real time. Cara menggunakan debugger adalah dengan menekan ikon bergambar serangga atau tekan `shift+f9`.

Terdapat beberapa fitur penting pada debugger, antara lain:

- **Step Into (F7):** untuk masuk ke dalam method yang sedang dipanggil.
- **Step Over (F8):** untuk melewati method tanpa masuk ke dalamnya.
- **Step Out (Shift+F8):** untuk keluar dari method saat ini dan Kembali ke method pemanggil

SOAL 2

2. Jelaskan Application class dalam arsitektur aplikasi Android dan fungsinya

A. Pembahasan

Application class dalam arsitektur aplikasi Android merupakan komponen inti yang merepresentasikan keseluruhan siklus hidup aplikasi. Kelas ini digunakan sebagai titik awal ketika aplikasi pertama kali dijalankan, sebelum aktivitas atau komponen lain dipanggil. Fungsinya antara lain untuk menginisialisasi konfigurasi global seperti library eksternal, logging, dependency injection, atau setup awal lainnya yang diperlukan oleh berbagai bagian aplikasi. Selain itu, Application class juga dapat digunakan untuk menyimpan data atau state global yang dapat diakses dari berbagai komponen, meskipun penggunaan ini perlu hati-hati agar tidak menimbulkan kebocoran memori. Dengan memanfaatkan Application class, pengembang dapat memastikan bahwa inisialisasi penting dilakukan hanya sekali dan berlaku secara konsisten selama aplikasi berjalan.

MODUL 5 : CONNECT TO THE INTERNET

SOAL 1

1. Lanjutkan aplikasi Android yang sudah dibuat pada Modul 4 dengan menambahkan modifikasi sesuai ketentuan berikut:
    a. Gunakan networking library seperti Retrofit atau Ktor agar aplikasi dapat mengambil data dari remote API. Dalam penggunaan networking library, sertakan generic response untuk status dan error handling pada API dan Flow untuk data stream.
    b. Gunakan KotlinX Serialization sebagai library JSON.
    c. Gunakan library seperti Coil atau Glide untuk image loading.
    d. API yang digunakan pada modul ini bebas, contoh API gratis The Movie Database (TMDB) API yang menampilkan data film. Berikut link dokumentasi API: https://developer.themoviedb.org/docs/getting-started
    e. Implementasikan konsep data persistence (misalnya offline-first app, pengaturan dark/light mode, fitur favorite, dll)
    f. Gunakan caching strategy pada Room..
    g. Untuk Modul 5, bebas memilih UI yang ingin digunakan, antara berbasis XML atau Jetpack Compose.

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

A. Source Code

1. AppDatabase.kt

Tabel 19. Source Code AppDatabase.kt

```
1  package com.allano.alquran.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 = [SurahEntity::class,
   AyahEntity::class], version = 4, exportSchema = false)
```

```
9  abstract class AppDatabase : RoomDatabase() {
10     abstract fun surahDao(): SurahDao
11     abstract fun ayahDao(): AyahDao
12
13     companion object {
14         @Volatile
15         private var INSTANCE: AppDatabase? = null
16
17         fun getDatabase(context: Context): AppDatabase {
18             return INSTANCE ?: synchronized(this) {
19                 val instance = Room.databaseBuilder(
20                     context.applicationContext,
21                     AppDatabase::class.java,
22                     "quran_database"
23                 ).fallbackToDestructiveMigration().build()
24                 INSTANCE = instance
25                 instance
26             }
27         }
28     }
29 }
```

2. AyahDao.kt

Tabel 20. Source Code AyahDao.kt

```
1  package com.allano.alquran.data.local
2
3  import androidx.room.Dao
4  import androidx.room.Insert
5  import androidx.room.OnConflictStrategy
6  import androidx.room.Query
7  import kotlinx.coroutines.flow.Flow
8
9  @Dao
10 interface AyahDao {
11
12     @Query("SELECT * FROM ayahs WHERE surahNumber =
   :surahNumber ORDER BY numberInSurah ASC")
13     fun getAyahsBySurahNumber(surahNumber: Int):
   Flow<List<AyahEntity>>
14
15     @Insert(onConflict = OnConflictStrategy.REPLACE)
16     suspend fun insertAll(ayahs: List<AyahEntity>)
17
18 }
```

3. AyahEntity.kt

Tabel 21. Source Code AyahEntity.kt

```
1  package com.allano.alquran.data.local
2
3  import androidx.room.Entity
4  import androidx.room.ForeignKey
5  import androidx.room.Index
6  import androidx.room.PrimaryKey
7
8  @Entity(
9      tableName = "ayahs",
10     foreignKeys = [
11         ForeignKey(
12             entity = SurahEntity::class,
13             parentColumns = ["number"],
14             childColumns = ["surahNumber"],
15             onDelete = ForeignKey.CASCADE
16         )
17     ],
18     indices = [Index(value = ["surahNumber"])]
19 )
20 data class AyahEntity(
21     @PrimaryKey
22     val number: Int,
23     val surahNumber: Int,
24     val numberInSurah: Int,
25     val arabicText: String,
26     val englishText: String
27 )
```

4. SurahDao.kt

Tabel 22. Source Code SurahDao.kt

```
1  package com.allano.alquran.data.local
2
3  import androidx.room.Dao
4  import androidx.room.Insert
5  import androidx.room.OnConflictStrategy
6  import androidx.room.Query
7  import kotlinx.coroutines.flow.Flow
8
9  @Dao
10 interface SurahDao {
11     @Query("SELECT * FROM surahs")
12     fun getAllSurahs(): Flow<List<SurahEntity>>
13
14     @Insert(onConflict = OnConflictStrategy.REPLACE)
15     suspend fun insertAll(surahs: List<SurahEntity>)
```

```
16
17      @Query("DELETE FROM surahs")
18      suspend fun deleteAll()
19  }
```

5. SurahEntity.kt

Tabel 23. Source Code SurahEntity.kt

```
1   package com.allano.alquran.data.local
2
3   import androidx.room.Entity
4   import androidx.room.PrimaryKey
5
6   @Entity(tableName = "surahs")
7   data class SurahEntity(
8       @PrimaryKey val number: Int,
9       val name: String,
10      val englishName: String,
11      val englishNameTranslation: String,
12      val revelationType: String,
13      val numberOfAyahs: Int
14  )
```

6. Ayah.kt

Tabel 24. Source Code Ayah.kt

```
1   @file:OptIn(InternalSerializationApi::class)
2
3   package com.allano.alquran.data.model
4
5   import kotlinx.serialization.InternalSerializationApi
6   import kotlinx.serialization.Serializable
7
8   @Serializable
9   data class Ayah(
10      val number: Int,
11      val numberInSurah: Int,
12      val text: String,
13      val juz: Int
14  )
15
16  @Serializable
17  data class SurahDetail(
18      val number: Int,
19      val name: String,
20      val englishName: String,
21      val englishNameTranslation: String,
22      val revelationType: String,
```

```
23      val ayahs: List<Ayah>
24  )
25
26  @Serializable
27  data class MultiEditionSurahDetailResponse(
28      val code: Int,
29      val status: String,
30      val data: List<SurahDetail>
31  )
32
33  data class MergedAyah(
34      val numberInSurah: Int,
35      val arabicText: String,
36      val englishText: String
37  )
```

7. Surah.kt

Tabel 25. Source Code Surah.kt

```
1   @file:OptIn(InternalSerializationApi::class)
2
3   package com.allano.alquran.data.model
4
5   import kotlinx.serialization.InternalSerializationApi
6   import kotlinx.serialization.SerialName
7   import kotlinx.serialization.Serializable
8   import kotlin.OptIn
9
10
11  @Serializable
12  data class Surah(
13      val number: Int,
14      val name: String,
15      @SerialName("englishName")
16      val englishName: String,
17      @SerialName("englishNameTranslation")
18      val englishNameTranslation: String,
19      @SerialName("revelationType")
20      val revelationType: String,
21      val numberOfAyahs: Int
22  )
23
24  @Serializable
25  data class SurahResponse(
26      val code: Int,
27      val status: String,
28      val data: List<Surah>
29  )
```

```
30
31 sealed class ApiResponse<out T> {
32     data class Success<out T>(val data: T) :
   ApiResponse<T>()
33     data class Error(val errorMessage: String) :
   ApiResponse<Nothing>()
34 }
```

8. ApiClient.kt

Tabel 26. Source Code ApiClient.kt

```
1  package com.allano.alquran.data.network
2
3  import
   com.jakewharton.retrofit2.converter.kotlinx.serialization.asConverterFactory
4  import kotlinx.serialization.json.Json
5  import okhttp3.OkHttpClient
6  import okhttp3.logging.HttpLoggingInterceptor
7  import okhttp3.MediaType.Companion.toMediaType
8  import retrofit2.Retrofit
9  import java.util.concurrent.TimeUnit
10
11 object ApiClient {
12     private const val BASE_URL = "https://api.alquran.cloud/"
13
14     private val json = Json { ignoreUnknownKeys = true }
15
16     private val loggingInterceptor = HttpLoggingInterceptor().apply {
17         level = HttpLoggingInterceptor.Level.BODY
18     }
19
20     private val client = OkHttpClient.Builder()
21         .addInterceptor(loggingInterceptor)
22         .connectTimeout(30, TimeUnit.SECONDS)
23         .readTimeout(30, TimeUnit.SECONDS)
24         .build()
25
26     val apiService: ApiService by lazy {
27         Retrofit.Builder()
28             .baseUrl(BASE_URL)
   .addConverterFactory(json.asConverterFactory("application/json".toMediaType()))
29             .client(client)
30             .build()
31             .create(ApiService::class.java)
32     }
33 }
```

9. ApiService.kt

Tabel 27. Source Code ApiService.kt

```
1  package com.allano.alquran.data.network
2
3  import com.allano.alquran.data.model.SurahResponse
4  import
   com.allano.alquran.data.model.MultiEditionSurahDetailResponse
5  import retrofit2.Response
6  import retrofit2.http.GET
7  import retrofit2.http.Path
8
9
10 interface ApiService {
11     @GET("v1/surah")
12     suspend fun getAllSurahs(): Response<SurahResponse>
13
14     @GET("v1/surah/{number}/editions/quran-uthmani,en.asad")
15     suspend fun getSurahDetailWithMultipleEditions(
16         @Path("number") surahNumber: Int
17     ): Response<MultiEditionSurahDetailResponse>
18
19 }
```

10. SurahRepository.kt

Tabel 28. Source Code SurahRepository.kt

```
1  package com.allano.alquran.data
2
3  import android.util.Log
4  import com.allano.alquran.data.local.AyahDao
5  import com.allano.alquran.data.local.AyahEntity
6  import com.allano.alquran.data.local.SurahDao
7  import com.allano.alquran.data.local.SurahEntity
8  import com.allano.alquran.data.model.MergedAyah
9  import com.allano.alquran.data.model.Surah
10 import com.allano.alquran.data.network.ApiClient
11 import kotlinx.coroutines.flow.Flow
12
13 class SurahRepository(private val surahDao: SurahDao, private val
   ayahDao: AyahDao) {
14
15
16     fun getSurahsStream(): Flow<List<SurahEntity>> =
   surahDao.getAllSurahs()
17
18     suspend fun refreshSurahs() {
19         try {
20             Log.d("SurahRepository", "Mencoba mengambil data dari
```

```
    network...")
21              val response = ApiClient.apiService.getAllSurahs()
22
23              if (response.isSuccessful) {
24                  val surahsFromApi = response.body()?.data ?:
    emptyList()
25                  Log.d("SurahRepository", "Sukses mengambil
    ${surahsFromApi.size} surah dari API")
26                  if (surahsFromApi.isNotEmpty()) {
27                      val surahEntities = surahsFromApi.map {
    it.toEntity() }
28                      surahDao.insertAll(surahEntities)
29                      Log.d("SurahRepository", "Data berhasil disimpan
    ke Room.")
30                  }
31              } else {
32                  Log.e("SurahRepository", "API Error:
    ${response.code()} ${response.message()}")
33              }
34          } catch (e: Exception) {
35              Log.e("SurahRepository", "Network Error: ${e.message}",
    e)
36          }
37      }
38      fun getAyahStream(surahNumber: Int): Flow<List<AyahEntity>> {
39          return ayahDao.getAyahsBySurahNumber(surahNumber)
40      }
41      suspend fun refreshSurahDetail(surahNumber: Int) {
42          try {
43              val response =
    ApiClient.apiService.getSurahDetailWithMultipleEditions(surahNumber)
44              if (response.isSuccessful) {
45                  val responseData = response.body()?.data
46                  val arabicEdition = responseData?.getOrNull(0)
47                  val englishEdition = responseData?.getOrNull(1)
48
49                  if (arabicEdition != null && englishEdition != null)
    {
50                      val mergedAyahs =
    arabicEdition.ayahs.zip(englishEdition.ayahs).map { (arabicAyah,
    englishAyah) ->
51                          AyahEntity(
52                              number = arabicAyah.number,
53                              surahNumber = surahNumber,
54                              numberInSurah =
    arabicAyah.numberInSurah,
55                              arabicText = arabicAyah.text,
```

```
56                                  englishText = englishAyah.text
57                              )
58                          }
59                          ayahDao.insertAll(mergedAyahs)
60                          Log.d("SurahRepository", "Detail ayat untuk
    surah $surahNumber disimpan ke DB.")
61                      }
62                  }
63              } catch (e: Exception) {
64                  Log.e("SurahRepository", "Gagal me-refresh detail surah:
    ", e)
65              }
66          }
67          suspend fun getSurahDetailFromApi(surahNumber: Int):
    List<MergedAyah>? {
68              Log.d("RepoDetail", "Memulai pengambilan detail untuk surah:
    $surahNumber")
69              return try {
70                  val response =
    ApiClient.apiService.getSurahDetailWithMultipleEditions(surahNumber)
71                  Log.d("RepoDetail", "Panggilan API berhasil
    (isSuccessful): ${response.isSuccessful}")
72
73                  if (response.isSuccessful) {
74                      val responseData = response.body()?.data
75                      Log.d("RepoDetail", "Jumlah edisi yang diterima dari
    API: ${responseData?.size}")
76
77                      val arabicEdition = responseData?.getOrNull(0)
78                      val englishEdition = responseData?.getOrNull(1)
79                      Log.d("RepoDetail", "Edisi Arab ditemukan:
    ${arabicEdition != null}")
80                      Log.d("RepoDetail", "Edisi Inggris ditemukan:
    ${englishEdition != null}")
81
82                      if (arabicEdition != null && englishEdition != null)
    {
83                          val mergedList =
    arabicEdition.ayahs.zip(englishEdition.ayahs).map { (arabicAyah,
    englishAyah) ->
84                              MergedAyah(
85                                  numberInSurah =
    arabicAyah.numberInSurah,
86                                  arabicText = arabicAyah.text,
87                                  englishText = englishAyah.text
88                              )
89                          }
```

```
90                      Log.d("RepoDetail", "Berhasil menggabungkan
    ${mergedList.size} ayat.")
91                      return mergedList
92                  } else {
93                      Log.e("RepoDetail", "Salah satu atau kedua edisi
    (Arab/Inggris) tidak ditemukan dalam respons API.")
94                      return null
95                  }
96              } else {
97                  Log.e("RepoDetail", "Panggilan API tidak sukses.
    Kode: ${response.code()}")
98                  return null
99              }
100         } catch (e: Exception) {
101             Log.e("RepoDetail", "Terjadi EXCEPTION saat mengambil
    detail: ", e)
102             return null
103         }
104     }
105     private fun Surah.toEntity(): SurahEntity {
106         return SurahEntity(
107             number = this.number,
108             name = this.name,
109             englishName = this.englishName,
110             englishNameTranslation = this.englishNameTranslation,
111             revelationType = this.revelationType,
112             numberOfAyahs = this.numberOfAyahs
113         )
114     }
115 }
```

11. AyahAdapter.kt

Tabel 29. Source Code AyahAdapter.kt

```
1   package com.allano.alquran.ui
2
3   import android.view.LayoutInflater
4   import android.view.ViewGroup
5   import androidx.recyclerview.widget.DiffUtil
6   import androidx.recyclerview.widget.ListAdapter
7   import androidx.recyclerview.widget.RecyclerView
8   import com.allano.alquran.data.local.AyahEntity
9   import com.allano.alquran.databinding.ItemAyahBinding
10
11  class AyahAdapter : ListAdapter<AyahEntity,
    AyahAdapter.AyahViewHolder>(AyahDiffCallback()) {
12
13      override fun onCreateViewHolder(parent: ViewGroup,
```

```
    viewType: Int): AyahViewHolder {
14          val binding =
    ItemAyahBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
15          return AyahViewHolder(binding)
16      }
17
18      override fun onBindViewHolder(holder: AyahViewHolder,
    position: Int) {
19          holder.bind(getItem(position))
20      }
21
22      class AyahViewHolder(private val binding:
    ItemAyahBinding) : RecyclerView.ViewHolder(binding.root) {
23          fun bind(ayahEntity: AyahEntity) {
24              binding.tvAyahNumber.text =
    ayahEntity.numberInSurah.toString()
25              binding.tvAyahTextArabic.text =
    ayahEntity.arabicText
26              binding.tvAyahTextEnglish.text =
    ayahEntity.englishText
27          }
28      }
29  }
30
31  class AyahDiffCallback : DiffUtil.ItemCallback<AyahEntity>()
    {
32      override fun areItemsTheSame(oldItem: AyahEntity,
    newItem: AyahEntity): Boolean {
33          return oldItem.number == newItem.number
34      }
35
36      override fun areContentsTheSame(oldItem: AyahEntity,
    newItem: AyahEntity): Boolean {
37          return oldItem == newItem
38      }
39  }
```

12. DetailFragment.kt

Tabel 30. Source Code DetailFragment.kt

```
1   package com.allano.alquran.ui
2
3   import android.os.Bundle
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
```

```
8  import androidx.core.view.isVisible
9  import androidx.fragment.app.Fragment
10 import androidx.fragment.app.activityViewModels
11 import androidx.fragment.app.viewModels
12 import androidx.lifecycle.lifecycleScope
13 import androidx.navigation.fragment.navArgs
14 import androidx.recyclerview.widget.LinearLayoutManager
15 import
   com.allano.alquran.databinding.FragmentDetailBinding
16 import kotlinx.coroutines.flow.collectLatest
17 import kotlinx.coroutines.launch
18
19 class DetailFragment : Fragment() {
20
21     private var _binding: FragmentDetailBinding? = null
22     private val binding get() = _binding!!
23
24     private val args: DetailFragmentArgs by navArgs()
25
26     private val detailViewModel: DetailViewModel by
   viewModels {
27
28 DetailViewModel.Factory(requireActivity().application,
   args.surahNumber)
29     }
30
31     private val surahViewModel: SurahViewModel by
   activityViewModels()
32
33     private lateinit var ayahAdapter: AyahAdapter
34
35     override fun onCreateView(
36         inflater: LayoutInflater, container: ViewGroup?,
37         savedInstanceState: Bundle?
       ): View {
38         _binding =
   FragmentDetailBinding.inflate(inflater, container, false)
39         return binding.root
40     }
41
42     override fun onViewCreated(view: View,
   savedInstanceState: Bundle?) {
43         super.onViewCreated(view, savedInstanceState)
44
45         setupInfoAwal()
46         setupRecyclerView()
47         observeAyahs()
```

```
48      }
49
50      private fun setupInfoAwal() {
51          val surah = surahViewModel.surahs.value.find {
    it.number == args.surahNumber }
52          surah?.let {
53              binding.tvDetailSurahName.text =
    it.englishName
54              binding.tvDetailSurahArabicName.text =
    it.name
55              binding.tvDetailTranslation.text =
    "\"${it.englishNameTranslation}\""
56              binding.tvDetailInfo.text = "Surah No:
    ${it.number} • ${it.revelationType} • ${it.numberOfAyahs}
    Verses"
57
58          }
59      }
60
61      private fun setupRecyclerView() {
62          ayahAdapter = AyahAdapter()
63          binding.rvAyahs.apply {
64              adapter = ayahAdapter
65              layoutManager =
    LinearLayoutManager(requireContext())
66          }
67      }
68
69      private fun observeAyahs() {
70          viewLifecycleOwner.lifecycleScope.launch {
71              detailViewModel.ayahs.collectLatest {
    ayahsList ->
72                  Log.d("DetailFragmentUI", "Menerima
    update dengan ${ayahsList.size} ayat.")
73
74                  binding.detailProgressBar.isVisible =
    ayahsList.isEmpty()
75
76                  ayahAdapter.submitList(ayahsList)
77              }
78          }
79      }
80
81      override fun onDestroyView() {
81          super.onDestroyView()
83          _binding = null
84
```

```
85      }
    }
```

13. DetailViewModel.kt

Tabel 31. Source Code DetailViewModel.kt

```
1   package com.allano.alquran.ui
2
3   import android.app.Application
4   import androidx.lifecycle.*
5   import com.allano.alquran.data.SurahRepository
6   import com.allano.alquran.data.local.AppDatabase
7   import com.allano.alquran.data.local.AyahEntity
8   import kotlinx.coroutines.flow.SharingStarted
9   import kotlinx.coroutines.flow.StateFlow
10  import kotlinx.coroutines.flow.stateIn
11  import kotlinx.coroutines.launch
12
13  class DetailViewModel(application: Application, private
    val surahNumber: Int) : AndroidViewModel(application) {
14
15      private val repository: SurahRepository
16
17      val ayahs: StateFlow<List<AyahEntity>>
18
19      init {
20          val surahDao =
    AppDatabase.getDatabase(application).surahDao()
21          val ayahDao =
    AppDatabase.getDatabase(application).ayahDao()
22          repository = SurahRepository(surahDao, ayahDao)
23
24          ayahs =
    repository.getAyahStream(surahNumber).stateIn(
25              scope = viewModelScope,
26              started =
    SharingStarted.WhileSubscribed(5000),
27              initialValue = emptyList()
28          )
29
30          viewModelScope.launch {
31              repository.refreshSurahDetail(surahNumber)
32          }
33      }
34
35      class Factory(private val app: Application, private
    val surahNumber: Int) : ViewModelProvider.Factory {
36          override fun <T : ViewModel> create(modelClass:
```

```
    Class<T>): T {
37              if
    (modelClass.isAssignableFrom(DetailViewModel::class.java))
    {
38                  @Suppress("UNCHECKED_CAST")
39                  return DetailViewModel(app, surahNumber)
    as T
40              }
41              throw IllegalArgumentException("Unknown
    ViewModel class")
42          }
43      }
44  }
```

14. HomeFragment.kt

Tabel 32. Source Code HomeFragment.kt

```
1   package com.allano.alquran.ui
2
3   import android.os.Bundle
4   import android.util.Log
5   import android.view.LayoutInflater
6   import android.view.View
7   import android.view.ViewGroup
8   import androidx.core.view.isVisible
9   import androidx.fragment.app.Fragment
10  import androidx.lifecycle.lifecycleScope
11  import androidx.navigation.fragment.findNavController
12  import androidx.recyclerview.widget.LinearLayoutManager
13  import kotlinx.coroutines.flow.collectLatest
14  import kotlinx.coroutines.launch
15  import com.allano.alquran.databinding.FragmentHomeBinding
16  import androidx.fragment.app.activityViewModels
17
18  class HomeFragment : Fragment() {
19
20      private var _binding: FragmentHomeBinding? = null
21      private val binding get() = _binding!!
22
23      private val viewModel: SurahViewModel by activityViewModels {
24          SurahViewModel.Factory(requireActivity().application)
25      }
26
27      override fun onCreateView(
28          inflater: LayoutInflater, container: ViewGroup?,
29          savedInstanceState: Bundle?
30      ): View {
```

```
31          _binding = FragmentHomeBinding.inflate(inflater, container,
    false)
32          return binding.root
33      }
34
35      override fun onViewCreated(view: View, savedInstanceState:
    Bundle?) {
36          super.onViewCreated(view, savedInstanceState)
37
38          val surahAdapter = SurahAdapter(viewModel)
39          binding.recyclerView.apply {
40              layoutManager = LinearLayoutManager(context)
41              adapter = surahAdapter
42          }
43
44          viewLifecycleOwner.lifecycleScope.launch {
45              binding.progressBar.isVisible = true
46
47              viewModel.surahs.collectLatest { surahs ->
48                  Log.d("HomeFragment", "Updating UI with
    ${surahs.size} surahs.")
49
50                  binding.progressBar.isVisible = false
51
52                  if (surahs.isNotEmpty()) {
53                      binding.tvError.isVisible = false
54                      surahAdapter.submitList(surahs)
55                  } else {
56                      binding.tvError.isVisible = true
57                  }
58              }
59          }
60
61          viewLifecycleOwner.lifecycleScope.launch {
62              viewModel.navigateToDetail.collect { surah ->
63                  surah?.let {
64                      val action =
    HomeFragmentDirections.actionHomeFragmentToDetailFragment(it.number)
65                      findNavController().navigate(action)
66                      viewModel.onDetailNavigated()
67                  }
68              }
69          }
70      }
71
72      override fun onDestroyView() {
73          super.onDestroyView()
```

```
74          _binding = null
75      }
76  }
```

15. SurahAdapter.kt

Tabel 33. Source Code SurahAdapter.kt

```
1   package com.allano.alquran.ui
2
3   import android.util.Log
4   import android.view.LayoutInflater
5   import android.view.ViewGroup
6   import androidx.recyclerview.widget.DiffUtil
7   import androidx.recyclerview.widget.ListAdapter
8   import androidx.recyclerview.widget.RecyclerView
9   import com.allano.alquran.data.local.SurahEntity
10  import com.allano.alquran.databinding.ItemSurahBinding
11
12  class SurahAdapter(private val viewModel: SurahViewModel) :
13      ListAdapter<SurahEntity,
    SurahAdapter.SurahViewHolder>(SurahDiffCallback()) {
14
15      override fun onCreateViewHolder(parent: ViewGroup,
    viewType: Int): SurahViewHolder {
16          val binding =
    ItemSurahBinding.inflate(LayoutInflater.from(parent.context),
    parent, false)
17          return SurahViewHolder(binding)
18      }
19
20      override fun onBindViewHolder(holder: SurahViewHolder,
    position: Int) {
21          val surah = getItem(position)
22          holder.bind(surah)
24          holder.itemView.setOnClickListener {
25              Log.d("SurahAdapter", "Item clicked:
    ${surah.englishName}")
26              viewModel.onSurahClicked(surah)
27          }
28      }
29
30      class SurahViewHolder(private val binding:
    ItemSurahBinding) :
31          RecyclerView.ViewHolder(binding.root) {
32          fun bind(surah: SurahEntity) {
33              binding.tvSurahNumber.text =
    surah.number.toString()
```

```
34              binding.tvSurahName.text = surah.englishName
35              binding.tvSurahTranslation.text =
    surah.englishNameTranslation
36              binding.tvSurahInfo.text =
    "${surah.revelationType} - ${surah.numberOfAyahs} verses"
37          }
38      }
39  }
40
41  class SurahDiffCallback :
    DiffUtil.ItemCallback<SurahEntity>() {
42      override fun areItemsTheSame(oldItem: SurahEntity,
    newItem: SurahEntity): Boolean {
43          return oldItem.number == newItem.number
44      }
45
46      override fun areContentsTheSame(oldItem: SurahEntity,
    newItem: SurahEntity): Boolean {
47          return oldItem == newItem
48      }
49  }
```

16. SurahViewModel.kt

Tabel 34. Source Code SurahViewModel.kt

```
1   package com.allano.alquran.ui
2
3   import android.app.Application
4   import android.util.Log
5   import androidx.lifecycle.*
6   import com.allano.alquran.data.SurahRepository
7   import com.allano.alquran.data.local.AppDatabase
8   import com.allano.alquran.data.local.SurahEntity
9   import com.allano.alquran.data.model.MergedAyah
10  import kotlinx.coroutines.flow.MutableStateFlow
11  import kotlinx.coroutines.flow.StateFlow
12  import kotlinx.coroutines.flow.catch
14  import kotlinx.coroutines.launch
15
16  class SurahViewModel(application: Application) :
    AndroidViewModel(application) {
18
19      private val repository: SurahRepository
20
21      private val _surahs =
    MutableStateFlow<List<SurahEntity>>(emptyList())
22      val surahs: StateFlow<List<SurahEntity>> = _surahs
```

```
23
24      private val _navigateToDetail =
    MutableStateFlow<SurahEntity?>(null)
25      val navigateToDetail: StateFlow<SurahEntity?> =
    _navigateToDetail
26
27      init {
28          val database =
    AppDatabase.getDatabase(application)
29          val surahDao = database.surahDao()
30          val ayahDao = database.ayahDao()
31          repository = SurahRepository(surahDao, ayahDao)
32          listenToDatabase()
33          refreshDataFromServer()
34      }
35
36      private fun listenToDatabase() {
37          viewModelScope.launch {
38              repository.getSurahsStream()
39                  .catch { e ->
40                      Log.e("SurahViewModel", "Error
    collecting from DB", e)
41                  }
42                  .collect { surahList ->
43                      _surahs.value = surahList
44                      Log.d("SurahViewModel", "Menerima
    ${surahList.size} surah dari database.")
45                  }
46          }
47      }
48
49      private fun refreshDataFromServer() {
50          viewModelScope.launch {
51              repository.refreshSurahs()
52          }
53      }
54
55      suspend fun getSurahDetailFromApi(surahNumber: Int):
    List<MergedAyah>? {
56          return
    repository.getSurahDetailFromApi(surahNumber)
57      }
58
59      fun onSurahClicked(surah: SurahEntity) {
60          _navigateToDetail.value = surah
61      }
62
```

```
63      fun onDetailNavigated() {
64          _navigateToDetail.value = null
65      }
66
67      class Factory(private val app: Application) :
    ViewModelProvider.Factory {
68          override fun <T : ViewModel> create(modelClass:
    Class<T>): T {
69              if
    (modelClass.isAssignableFrom(SurahViewModel::class.java))
70  {
71                  @Suppress("UNCHECKED_CAST")
72                  return SurahViewModel(app) as T
73              }
74              throw IllegalArgumentException("Unable to
    construct viewmodel")
75          }
76      }
77  }
```

17. MainActivity.kt

Tabel 35. Source Code MainActivity.kt

```
1   package com.allano.alquran
2
3   import androidx.appcompat.app.AppCompatActivity
4   import android.os.Bundle
5
6   class MainActivity : AppCompatActivity() {
7       override fun onCreate(savedInstanceState: Bundle?) {
8           super.onCreate(savedInstanceState)
9           setContentView(R.layout.activity_main)
10      }
11  }
```

18. activity_main.xml

Tabel 36. Source Code activity_main.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:id="@+id/nav_host_fragment"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7
    android:name="androidx.navigation.fragment.NavHostFragment"
8       app:defaultNavHost="true"
9
```

```
10      app:navGraph="@navigation/nav_graph"
        />
```

19. fragment_detail.xml

Tabel 37. Source Code fragment_detail.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.core.widget.NestedScrollView
    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=".ui.DetailFragment">
8
9       <LinearLayout
10          android:layout_width="match_parent"
11          android:layout_height="wrap_content"
12          android:orientation="vertical">
13
14          <androidx.constraintlayout.widget.ConstraintLayout
15              android:layout_width="match_parent"
16              android:layout_height="wrap_content"
17              android:padding="24dp">
18
19              <TextView
20                  android:id="@+id/tv_detail_surah_name"
21                  android:layout_width="0dp"
22                  android:layout_height="wrap_content"
23
    android:textAppearance="?attr/textAppearanceHeadline4"
24                  android:textStyle="bold"
25                  app:layout_constraintEnd_toEndOf="parent"
26
    app:layout_constraintStart_toStartOf="parent"
27                  app:layout_constraintTop_toTopOf="parent"
28                  tools:text="Al-Fatiha" />
29
30              <TextView
31
    android:id="@+id/tv_detail_surah_arabic_name"
32                  android:layout_width="0dp"
33                  android:layout_height="wrap_content"
34                  android:layout_marginTop="8dp"
35
    android:textAppearance="?attr/textAppearanceHeadline5"
36                  app:layout_constraintEnd_toEndOf="parent"
37
```

```
    app:layout_constraintStart_toStartOf="parent"
38
    app:layout_constraintTop_toBottomOf="@+id/tv_detail_surah_
    name"
39                  tools:text="سُورَةُ ٱلْفَـاتِـحَةِ" />
40
41              <TextView
42                  android:id="@+id/tv_detail_translation"
43                  android:layout_width="0dp"
44                  android:layout_height="wrap_content"
45                  android:layout_marginTop="4dp"
46
    android:textAppearance="?attr/textAppearanceBody1"
47                  android:textStyle="italic"
48                  app:layout_constraintEnd_toEndOf="parent"
49
    app:layout_constraintStart_toStartOf="parent"
50
    app:layout_constraintTop_toBottomOf="@+id/tv_detail_surah_
    arabic_name"
51                  tools:text="&amp;quot;The Opening&amp;quot;" />
52
53              <TextView
54                  android:id="@+id/tv_detail_info"
55                  android:layout_width="0dp"
56                  android:layout_height="wrap_content"
57                  android:layout_marginTop="16dp"
58
    android:textAppearance="?attr/textAppearanceBody2"
59
    android:textColor="?android:attr/textColorSecondary"
60                  app:layout_constraintEnd_toEndOf="parent"
61
    app:layout_constraintStart_toStartOf="parent"
62
    app:layout_constraintTop_toBottomOf="@+id/tv_detail_transl
    ation"
63                  tools:text="Surah No: 1 • Meccan • 7
    Verses" />
64
65
    </androidx.constraintlayout.widget.ConstraintLayout>
66
67          <ProgressBar
68              android:id="@+id/detail_progress_bar"
69              style="?android:attr/progressBarStyle"
70              android:layout_width="wrap_content"
```

```
71              android:layout_height="wrap_content"
72              android:layout_gravity="center_horizontal"
73              android:layout_marginTop="16dp"
74              android:visibility="visible"
75              tools:visibility="gone" />
76
77          <androidx.recyclerview.widget.RecyclerView
78              android:id="@+id/rv_ayahs"
79              android:layout_width="match_parent"
80              android:layout_height="wrap_content"
81              android:nestedScrollingEnabled="false"
82              tools:itemCount="5"
83              tools:listitem="@layout/item_ayah" />
84
85      </LinearLayout>
86  </androidx.core.widget.NestedScrollView>
```

20. fragment_home.xml

Tabel 38. Source Code fragment_home.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/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=".ui.HomeFragment">
8
9       <androidx.recyclerview.widget.RecyclerView
10          android:id="@+id/recycler_view"
11          android:layout_width="0dp"
12          android:layout_height="0dp"
13          android:clipToPadding="false"
14          android:paddingVertical="4dp"
15          android:contentDescription="FUCK THE BLINDS"
16          app:layout_constraintBottom_toBottomOf="parent"
17          app:layout_constraintEnd_toEndOf="parent"
18          app:layout_constraintStart_toStartOf="parent"
19          app:layout_constraintTop_toTopOf="parent" />
20
21      <ProgressBar
22          android:id="@+id/progress_bar"
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          android:contentDescription="Fuck the blinds"
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="Loading."
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
46  </androidx.constraintlayout.widget.ConstraintLayout>
```

21. item_ayah.xml

Tabel 39. Source Code item_ayah.xml

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:tools="http://schemas.android.com/tools"
4       android:layout_width="match_parent"
5       android:layout_height="wrap_content"
6       android:orientation="vertical"
7       android:paddingHorizontal="16dp"
8       android:paddingVertical="12dp">
9
10      <TextView
11          android:id="@+id/tv_ayah_number"
12          android:layout_width="wrap_content"
13          android:layout_height="wrap_content"
14          android:layout_gravity="start"
15          android:paddingHorizontal="8dp"
16          android:paddingVertical="2dp"
17          android:textColor="?attr/colorPrimary"
18          android:textStyle="bold"
19          tools:text="1" />
20
21      <TextView
22          android:id="@+id/tv_ayah_text_arabic"
23          android:layout_width="match_parent"
24          android:layout_height="wrap_content"
```

```
25          android:layout_marginTop="8dp"
26          android:gravity="end"
27          android:lineSpacingMultiplier="1.5"
28      android:textAppearance="?attr/textAppearanceHeadline6"
29          android:textColor="?android:attr/textColorPrimary"
30          tools:text="بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ"
31          />
32
33      <TextView
34          android:id="@+id/tv_ayah_text_english"
35          android:layout_width="match_parent"
36          android:layout_height="wrap_content"
37          android:layout_marginTop="4dp"
38          android:lineSpacingMultiplier="1.2"
39          android:textAppearance="?attr/textAppearanceBody2"
40
    android:textColor="?android:attr/textColorSecondary"
41          tools:text="In the name of Allah, the Entirely
    Merciful, the Especially Merciful." />
42
43  </LinearLayout>
```

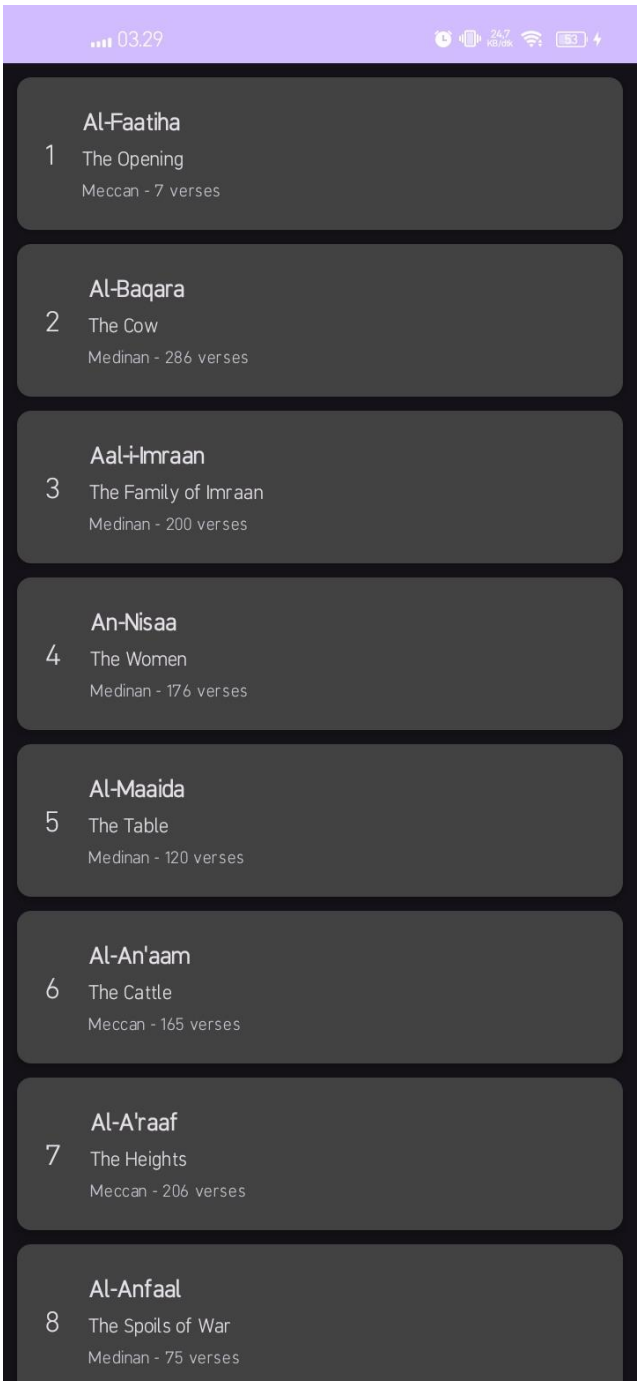
22. item_surah.xml

Tabel 40. Source Code item_surah.xml

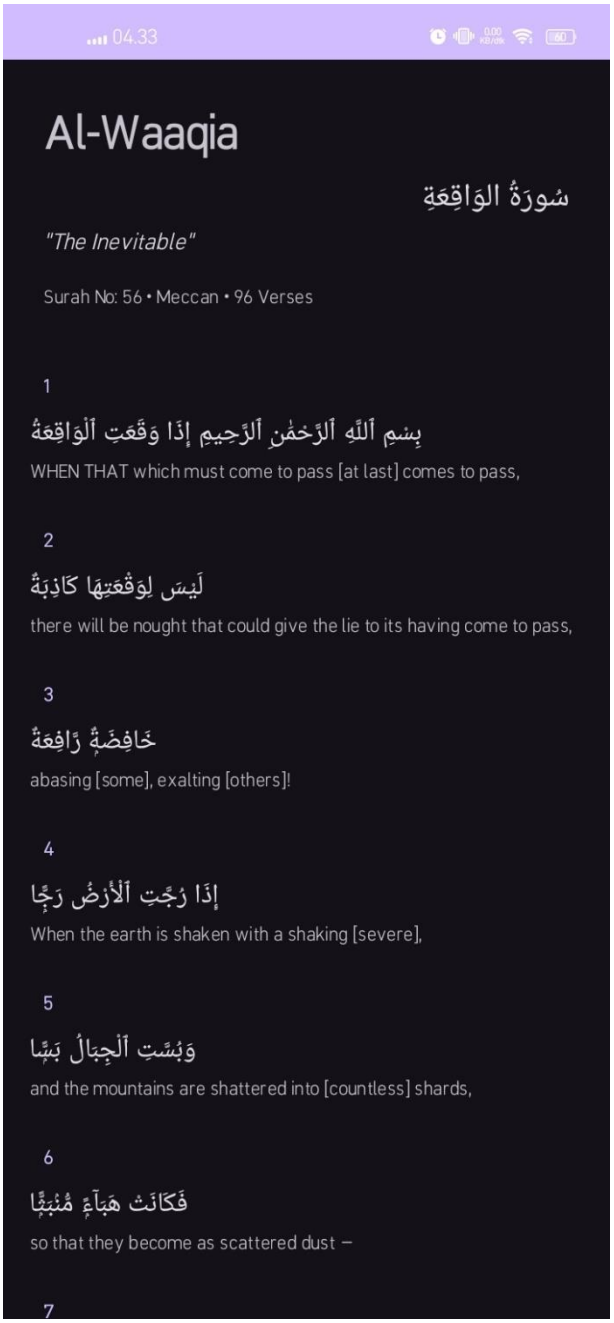
```
1   <?xml version="1.0" encoding="utf-8"?>
2   <androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="wrap_content"
7       android:layout_marginHorizontal="8dp"
8       android:layout_marginVertical="4dp"
9       app:cardCornerRadius="8dp"
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="16dp">
16
17          <TextView
18              android:id="@+id/tvSurahNumber"
19              android:layout_width="wrap_content"
20              android:layout_height="wrap_content"
21  android:textAppearance="?attr/textAppearanceHeadline6"
22              app:layout_constraintStart_toStartOf="parent"
```

```
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            tools:text="1" />

        <TextView
            android:id="@+id/tvSurahName"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginStart="16dp"
android:textAppearance="?attr/textAppearanceListItem"
            android:textStyle="bold"
            app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toEndOf="@id/tvSurahNumber"
            app:layout_constraintTop_toTopOf="parent"
            tools:text="Al-Fatiha" />

        <TextView
            android:id="@+id/tvSurahTranslation"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
android:textAppearance="?attr/textAppearanceBody2"
         app:layout_constraintEnd_toEndOf="@id/tvSurahName"
app:layout_constraintStart_toStartOf="@id/tvSurahName"
app:layout_constraintTop_toBottomOf="@id/tvSurahName"
            tools:text="The Opening" />

        <TextView
            android:id="@+id/tvSurahInfo"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="4dp"
android:textAppearance="?attr/textAppearanceCaption"
app:layout_constraintStart_toStartOf="@id/tvSurahName"
app:layout_constraintTop_toBottomOf="@id/tvSurahTranslation"
            tools:text="Meccan - 7 verses" />

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

B. Output Program



Gambar 10. Screenshot Halaman Utama



Gambar 11. Screenshot Halaman Detail

C. Pembahasan

1. SurahEntity.kt dan AyahEntity.kt:

File ini berfungsi sebagai skema untuk tabel dalam database SQLite. Anotasi @Entity memberitahu Room bahwa data class ini harus diubah menjadi tabel, setiap property di dalam class, seperti number, arabicText, dan atribut lainnya akan menjadi kolom dalam tabel tersebut. Anotasi @PrimaryKey menandakan kolom yang nilainya unik untuk setiap baris

(Unique Key) yang digunakan sebagai pengenal utama. Di AyahEntity.kt, didefinisikan @ForeignKey sebagai penghubung relasional ke tabel surah untuk menjaga integritas data

2. **SurahDao.kt dan AyahDao.kt:**

Data Access Object (DAO) adalah interface yang berisi daftar fungsi untuk berinteraksi dengan database. Dengan anotasi @Dao, room akan secara otomatis membuatkan implementasi dari setiap fungsi yang didefinisikan. Anotasi @Query berfungsi untuk menulis perintah SQL yang akan divalidasi oleh room. Anotasi @Insert dengan OnConflictStrategy.REPLACE akan menimpa data lama jika ada data baru dengan Primary Key yang sama. Semua fungsi Input/Output ditandai sebagai suspend agar bisa dipanggil coroutine tanpa memblokir thread utama, sementara fungsi yang mengembalikan Flow menyediakan stream data yang otomatis mengirimkan data terbaru ke subscribernya setiap ada perubahan di tabel.

3. **AppDatabase.kt**

Anotasi @Database mendaftarkan semua kelas @Entity yang menjadi bagian dari skema databse dan juga mendefinisikan nomor version dari database tersebut. Nomor versi wajib dinaikkan setiap kali ada perubahan di struktur tabel. Metode .fallbackToDestructioveMigration() memerintahkan Room untuk menghapus dan membuat ulang seluruh database jika terdeteksi kenaikan versi dan tidak ada skrip migrasi spesifik yang diberikan.

4. **Ayah.kt dan SurahDetail.kt**

File-file ini berisi data class yang strukturnya berbentuk JSON yang dikirimkan API. Anotasi @Serializabel dari library KotlinX Serialization memungkinkan proses deserialization yang mengkonversi teks JSON menjadi objek Kotlin yang bisa digunakan dalam kode ini. Anotasi @SerialName digunakan sebagai pemetaan jika nama field di JSON berbeda dengan nama property di kelas Kotlin

5. **ApiService.kt**

Fungsi didalamnya merepresentasikan satu endpoint API. Anotasi @GET mendefinisikan tipe request HTTP dan relative pathnya. Anotasi @Path digunakan untuk menyisipkan nilai dinamis ke URL seperti nomor surah. Seluruh fungsi didefinisikan sebagai suspend karena panggilan jaringan adalah operasi yang harus dijalankan diluar thread utama.

6. **ApiClient.kt**

File ini bertanggung jawab untuk membuat dan mengkonfigurasi satu instance Retrofit untuk seluruh aplikasi. Didalamnya didefinisikan baseUrl dari API, konfigurasi OkHttpClient, dan ConverterFactory yang memberitahu Retrofit bagaimana memparsing JSON menjadi objek Kotlin menggunakan Serialization.

7. **SurahRepository.kt:**

Repository bertindak sebagai mediator antar sumber data (jaringan dan lokal) dan bagian dari viewModel. SurahRepository mengimplementasikan pola offline-first dengan menyediakan

fungsi get…Stream yang mengembalikan Flow langsung dari DAO sebagai sumber Tunggal untuk UI, dan fungsi refresh yang merupakan fungsi suspend untuk mengambil data terbaru dari jaringan dan menyimpannya ke database lokal, yang kemudian akan diupdate secara otomatis.

8. **SurahViewModel.kt dan DetailViewModel.kt:**

ViewModel menyimpan dan mengelola data yang berhubungan dengan UI. Data didalamnya akan selamat dari perubahan konfigurasi, seperti perubahan rotasi layar. StateFlow digunakan untuk menampung state seperti daftar surah dan mengeksposnya ke UI. Operasi yang akan berjalan lama seperti meminta data ke repository akan dijalankan dalam viewModelScope, yang secara otomatis akan membatalkan semua tugasnya jika viewmodel dihancurkan.

9. **HomeFragment.kt dan DetailFragment.kt:**

Dalam onViewCreated, logika untuk menginisialisasi UI ditempatkan. ViewBinding digunakan untuk mengakses elemen dari XML secara aman. Fungsi utamanya adalam mengamati StateFlow dari Viewmodel yang menggunakan viewLifecycleOwner.lifecycleScope. setiap ada data baru dari ViewModel, blok collect akan dieksekusi untuk memperbarui tampilan. File-file ini juga mendeteksi interaksi pengguna seperti onClick dan meneruskannya ke viewModel.

10. **SurahAdapter.kt dan ListAdapter.kt:**

Adapter berguna sebagai jembatan data dan RecyclerView. ListAdapter di extend untuk menampilkan daftar data yang bisa berubah. ListAdapter menggunakan DiffUtil.ItemCallback untuk menghitung perbedaan antara data lama dan data baru lalu secara otomatis memperbaruinya. Didalamnya ada sebuah template yang memegang referensi ke item di file XML.

11. **Layout.xml**

File-file layout XML mendefinisikan tampilan visual aplikasi secara deklaratif seperti CSS. Fragment_home.xml dan fragment_detail.xml menampilkan struktur komponen untuk masing masing layar, sementara item_surah.xml dan item_ayah.xml mendefinisikan tampilan perbaris.

TAUTAN GIT

Berikut adalah tautan untuk semua source code yang telah dibuat.

https://github.com/AllanoLintang/pemro-mobile.git