

**LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 1**



ANDROID BASIC WITH KOTLIN

Oleh:

Muhammad Rizki Ramadhan

NIM. 2310817310008

**PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
APRIL 2025**

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN I
MODUL 1

Laporan Praktikum Pemrograman Mobile Modul 1: Android Basic with Kotlin ini disusun sebagai syarat lulus mata kuliah Praktikum Pemrograman Mobile. Laporan Praktikum ini dikerjakan oleh:

Nama Praktikan : Muhammad Rizki Ramadhan
NIM : 2310817310008

Menyetujui,
Asisten Praktikum

Mengetahui,
Dosen Penanggung Jawab Praktikum

Muhammad Raka Azwar
NIM. 2210817210012

Andreyan Rizky Baskara, S.Kom., M.Kom.
NIP. 19930703 20190301011

DAFTAR ISI

LEMBAR PENGESAHAN	1
DAFTAR ISI	2
DAFTAR GAMBAR.....	3
DAFTAR TABEL	4
SOAL 1	5
A. Source Code.....	8
B. Output Program	14
C. Pembahasan	20
D. Tautan Git	22

DAFTAR GAMBAR

Gambar 1. Tampilan Awal Aplikasi.....	5
Gambar 2. Tampilan Dadu Setelah Diroll	6
Gambar 3. Tampilan Roll Dadu Double.....	7
Gambar 4. Tampilan Awal Aplikasi XML	14
Gambar 5. Tampilan Dadu Setelah Di-Roll XML	15
Gambar 6. Tampilan Roll Dadu Double XML.....	16
Gambar 7. Tampilan Awal Aplikasi Jetpack Compose.....	17
Gambar 8. Tampilan Dadu Setelah Di-Roll Jetpack Compose	18
Gambar 9. Tampilan Roll Dadu Double Jetpack Compose.....	19

DAFTAR TABEL

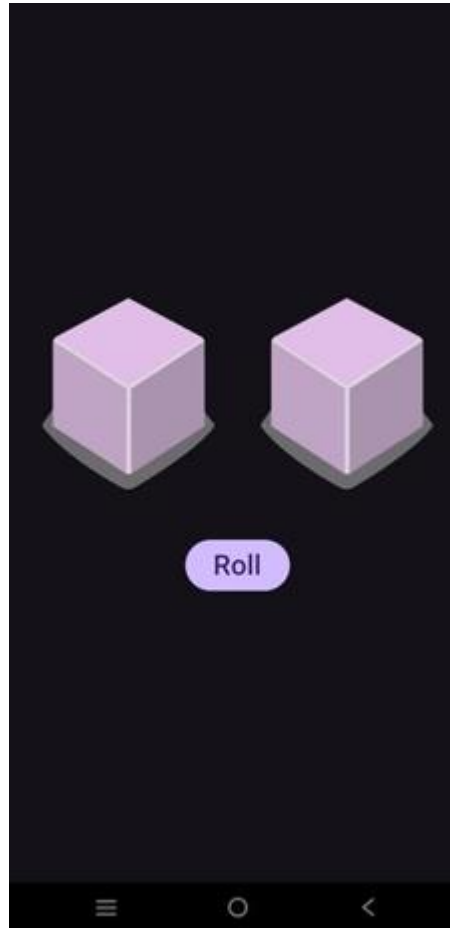
Table 1. Source Code Jawaban Soal 1 XML.....	9
Table 2. Source Code Jawaban Soal 1 XML.....	10
Table 3. Source Code Jawaban Soal 1 Jetpack Compose.....	13

SOAL 1

Soal Praktikum:

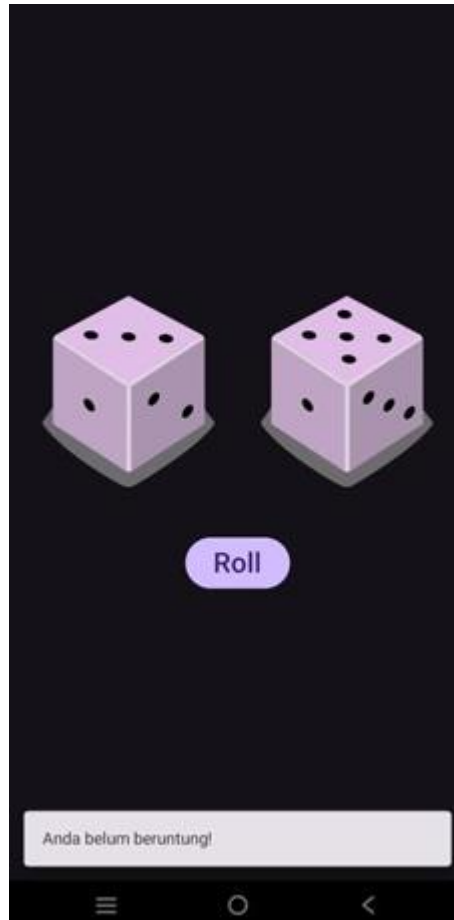
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



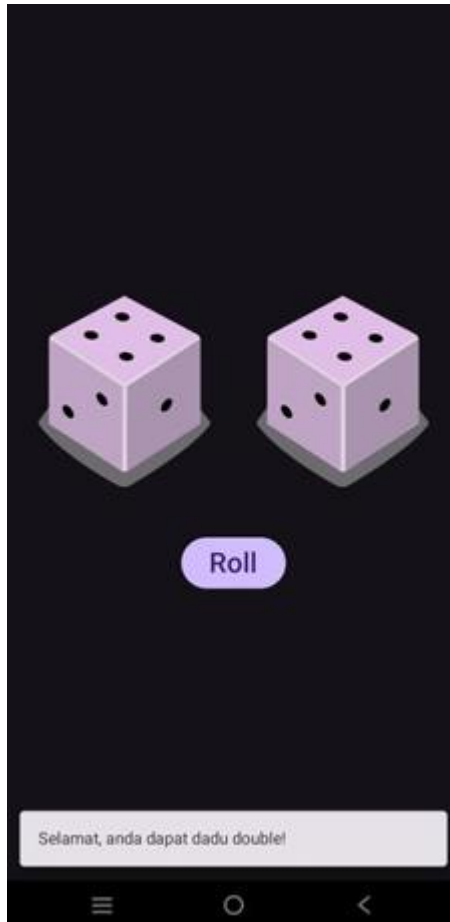
Gambar 1. Tampilan Awal Aplikasi

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



Gambar 2. Tampilan Dadu Setelah Diroll

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



Gambar 3. Tampilan Roll Dadu Double

4. Buatlah aplikasi tersebut menggunakan XML dan Jetpack Compose.
5. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 1 dalam bentuk project**. Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.
6. Untuk gambar dadu dapat didownload pada link berikut:
https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view?usp=sharing

A. Source Code

XML:

MainActivity.kt

```
1 package com.example.diceroll
2
3 import android.os.Bundle
4 import android.widget.Button
5 import android.widget.ImageView
6 import com.google.android.material.snackbar.Snackbar
7 import androidx.appcompat.app.AppCompatActivity
8 import com.example.diceroll2.R
9
10 class MainActivity : AppCompatActivity() {
11
12     private lateinit var diceImage1: ImageView
13     private lateinit var diceImage2: ImageView
14     private lateinit var rollButton: Button
15     private lateinit var coordinatorLayout:
16     androidx.coordinatorlayout.widget.CoordinatorLayout
17
18     private val diceImages = listOf(
19         R.drawable.dice_0,
20         R.drawable.dice_1,
21         R.drawable.dice_2,
22         R.drawable.dice_3,
23         R.drawable.dice_4,
24         R.drawable.dice_5,
25         R.drawable.dice_6
26     )
27
28     override fun onCreate(savedInstanceState: Bundle?) {
29         super.onCreate(savedInstanceState)
30         setContentView(R.layout.activity_main)
31
32         coordinatorLayout
33         findViewById(R.id.coordinatorLayout)
34         diceImage1 = findViewById(R.id.dice_image1)
35         diceImage2 = findViewById(R.id.dice_image2)
36         rollButton = findViewById(R.id.roll_button)
37
38         rollButton.setOnClickListener {
39             val result1 = (1..6).random()
40         }
41     }
42 }
```

38	val result2 = (1..6).random()
39	
40	diceImage1.setImageResource(diceImages[result1])
41	diceImage2.setImageResource(diceImages[result2])
42	
43	val message = if (result1 == result2) {
44	"Selamat, anda dapat dadu double!"
45	} else {
46	"Anda belum beruntung!"
47	}
48	
49	Snackbar.make(coordinatorLayout, message,
	Snackbar.LENGTH_SHORT).show()
50	}
51	}
52	}

Table 1. Source Code Jawaban Soal 1 XML

activity_main.xml

1	<?xml version="1.0" encoding="utf-8"?>
2	<androidx.coordinatorlayout.widget.CoordinatorLayout
3	xmlns:android="http://schemas.android.com/apk/res/android"
4	android:id="@+id/coordinatorLayout"
5	android:layout_width="match_parent"
6	android:layout_height="match_parent"
7	android:padding="24dp">
8	
9	<LinearLayout
10	android:orientation="vertical"
11	android:gravity="center"
12	android:layout_width="match_parent"
13	android:layout_height="match_parent">
14	
15	<LinearLayout
16	android:orientation="horizontal"
17	android:layout_width="wrap_content"
18	android:layout_height="wrap_content"
19	android:layout_marginBottom="16dp">
20	
21	<ImageView
22	android:id="@+id/dice_image1"
23	android:layout_width="150dp"
24	android:layout_height="match_parent"

25	android:src="@drawable/dice_0" />
26	
27	<ImageView
28	android:id="@+id/dice_image2"
29	android:layout_width="150dp"
30	android:layout_height="170dp"
31	android:src="@drawable/dice_0" />
32	</LinearLayout>
33	
34	<Button
35	android:id="@+id/roll_button"
36	android:layout_width="wrap_content"
37	android:layout_height="wrap_content"
38	android:text="Roll" />
39	</LinearLayout>
40	</androidx.coordinatorlayout.widget.CoordinatorLayout>

Table 2. Source Code Jawaban Soal 1 XML

Jetpack Compose:

MainActivity.kt

1	package com.example.diceroll
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.Image
8	import androidx.compose.foundation.layout.Arrangement
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.SnackbarDuration
19	import androidx.compose.material3.SnackbarHost
20	import androidx.compose.material3.SnackbarHostState
21	import androidx.compose.material3.Text
22	import androidx.compose.runtime.Composable

```

23 import androidx.compose.runtime.mutableStateOf
24 import androidx.compose.runtime.remember
25 import androidx.compose.runtime.getValue
26 import androidx.compose.runtime.rememberCoroutineScope
27 import androidx.compose.runtime.setValue
28 import androidx.compose.ui.Alignment
29 import androidx.compose.ui.Modifier
30 import androidx.compose.ui.res.painterResource
31 import androidx.compose.ui.res.stringResource
32 import androidx.compose.ui.tooling.preview.Preview
33 import androidx.compose.ui.unit.dp
34 import com.example.diceroll.ui.theme.DiceRollTheme
35 import kotlinx.coroutines.Job
36 import kotlinx.coroutines.launch
37
38 class MainActivity : ComponentActivity() {
39     override fun onCreate(savedInstanceState: Bundle?) {
40         super.onCreate(savedInstanceState)
41         enableEdgeToEdge()
42         setContent {
43             DiceRollTheme {
44                 DiceRollerApp()
45             }
46         }
47     }
48
49     @Preview
50     @Composable
51     fun DiceRollerApp() {
52         DiceWithButtonAndImage()
53     }
54
55     @Composable
56     fun DiceWithButtonAndImage(
57         modifier: Modifier = Modifier
58             .fillMaxSize()
59             .wrapContentSize(Alignment.Center)
60     ) {
61         var dismissJob by remember { mutableStateOf<Job?>(null) }
62
63         var result1 by remember { mutableStateOf(0) }
64         var result2 by remember { mutableStateOf(0) }
65         val snackbarHostState = remember { SnackbarHostState() }
66     }

```

```

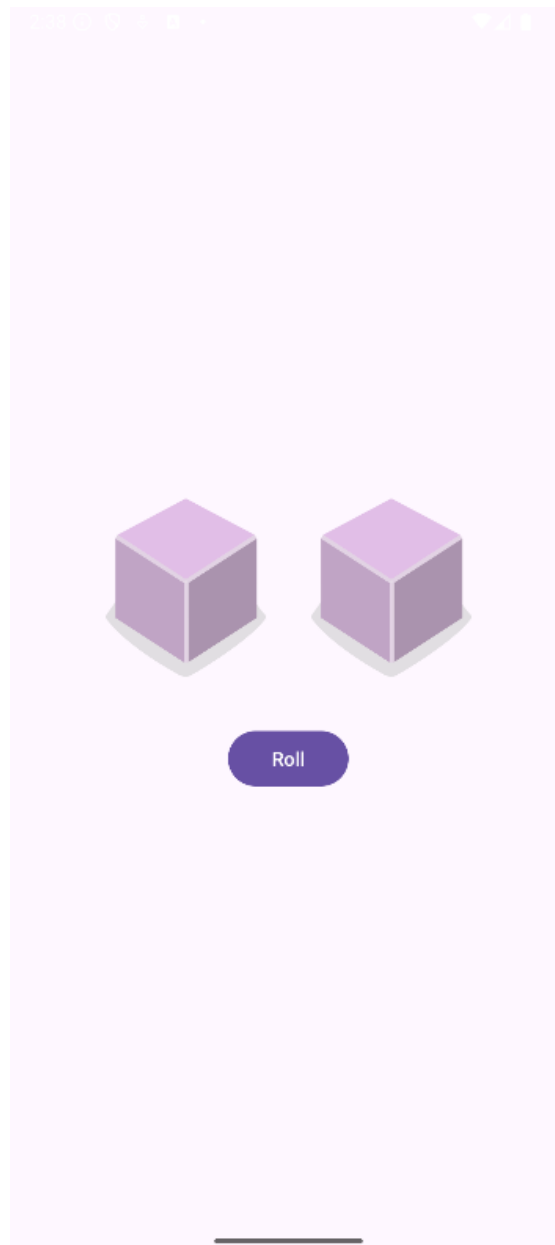
66         val coroutineScope = rememberCoroutineScope()
67
68         val dice1 = when (result1) {
69             0 -> R.drawable.dice_0
70             1 -> R.drawable.dice_1
71             2 -> R.drawable.dice_2
72             3 -> R.drawable.dice_3
73             4 -> R.drawable.dice_4
74             5 -> R.drawable.dice_5
75             else -> R.drawable.dice_6
76         }
77
78         val dice2 = when (result2) {
79             0 -> R.drawable.dice_0
80             1 -> R.drawable.dice_1
81             2 -> R.drawable.dice_2
82             3 -> R.drawable.dice_3
83             4 -> R.drawable.dice_4
84             5 -> R.drawable.dice_5
85             else -> R.drawable.dice_6
86         }
87
88         Scaffold(
89             snackbarHost = { SnackbarHost(snackbarHostState) }
90         ) { padding ->
91             Column(
92                 modifier = modifier.padding(padding),
93                 horizontalAlignment
94                 Alignment.CenterHorizontally
95             ) {
96                 Row(
97                     horizontalArrangement
98                     Arrangement.spacedBy(16.dp),
99                     verticalAlignment
100                     Alignment.CenterVertically
101                 ) {
102                     Image(
103                         painter = painterResource(dice1),
104                         contentDescription
105                         result1.toString()
106                     )
107                     Image(
108                         painter = painterResource(dice2),

```

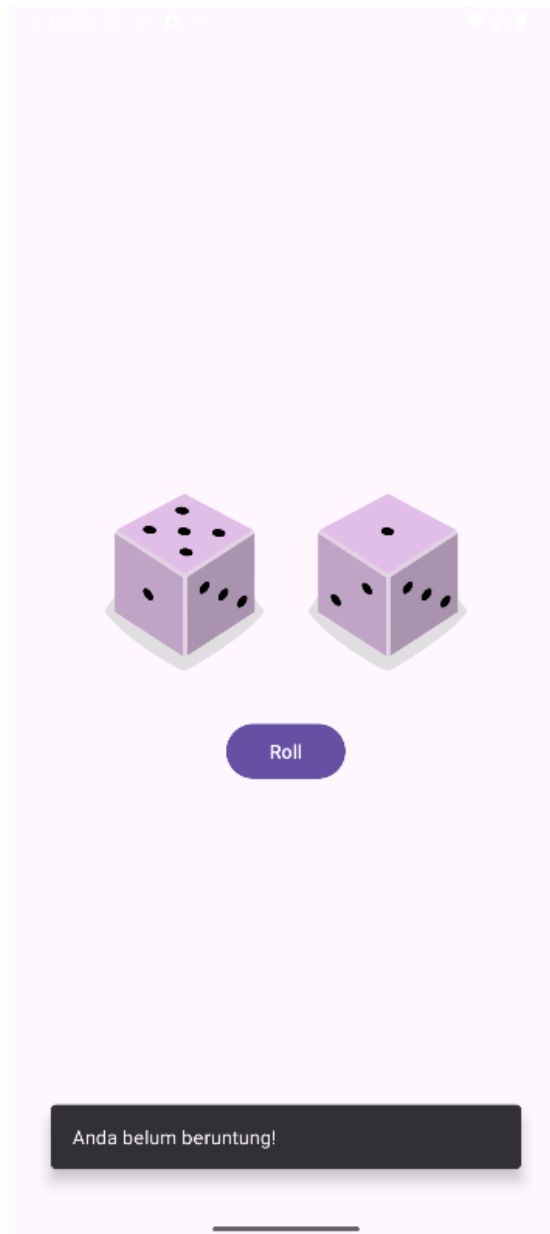
106	contentDescription	=
	result2.toString()	
107)	
108	}	
109		
110	Spacer(modifier = Modifier.height(10.dp))	
111		
112	Button(onClick = {	
113	result1 = (1..6).random()	
114	result2 = (1..6).random()	
115		
116	val message = if (result1 == result2) {	
117	"Salamat, anda dapat dadu double!"	
118	} else {	
119	"Anda belum beruntung!"	
120	}	
121		
122	coroutineScope.launch {	
123		
124	snackbarHostState.currentSnackbarData?.dismiss()	
125		
126	snackbarHostState.showSnackbar(
127	message = message,	
128	duration	=
	SnackbarDuration.Indefinite	
129)	
130	}	
131		
132	dismissJob?.cancel()	
133		
134	dismissJob = coroutineScope.launch {	
135	kotlinx.coroutines.delay(2000)	
136		
137	snackbarHostState.currentSnackbarData?.dismiss()	
138	}	
139	}) {	
140	Text(stringResource(R.string.roll))	
141	}	
142	}	
143	}	
144	}	
	}	

Table 3. Source Code Jawaban Soal 1 Jetpack Compose

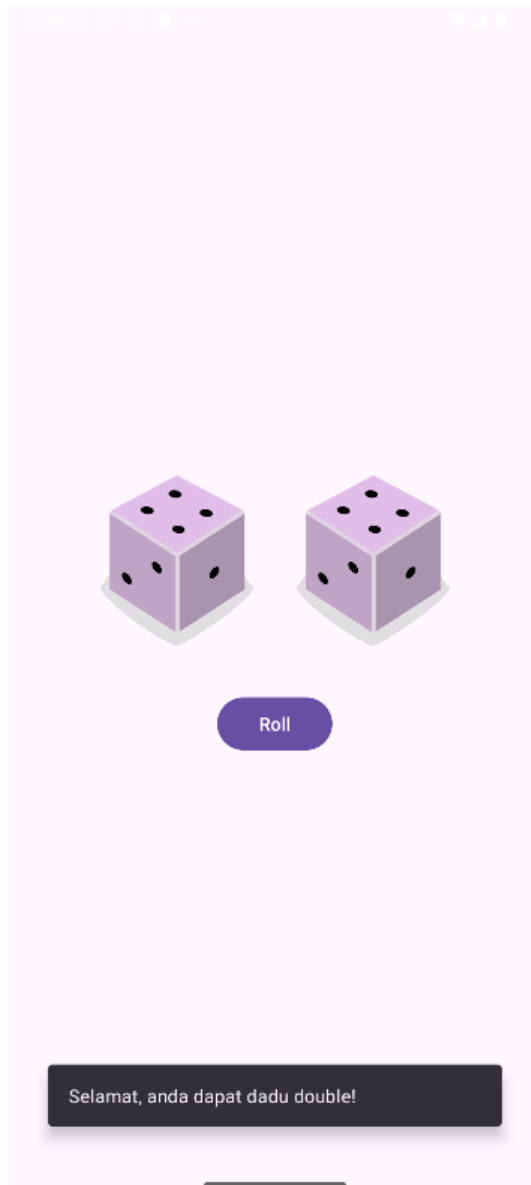
B. Output Program



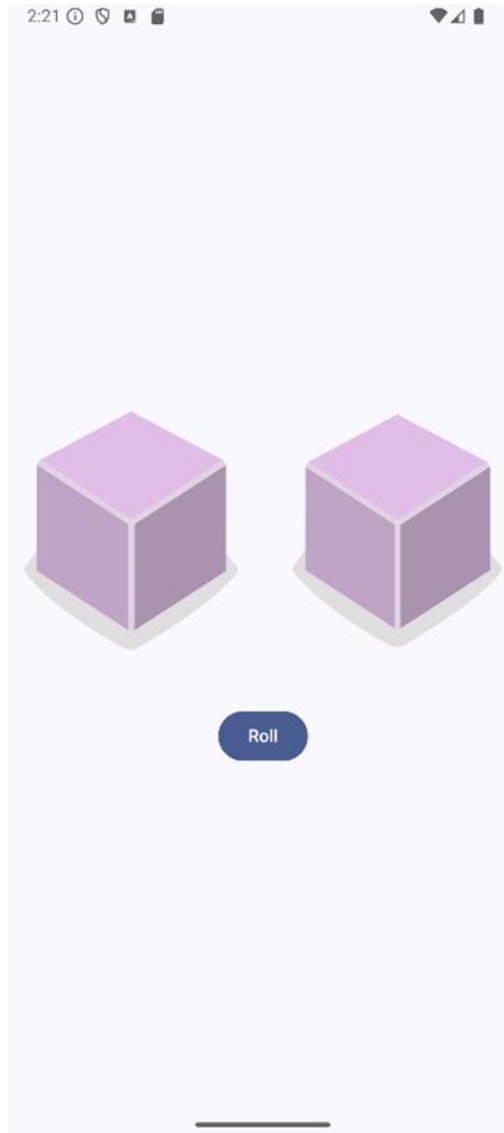
Gambar 4. Tampilan Awal Aplikasi XML



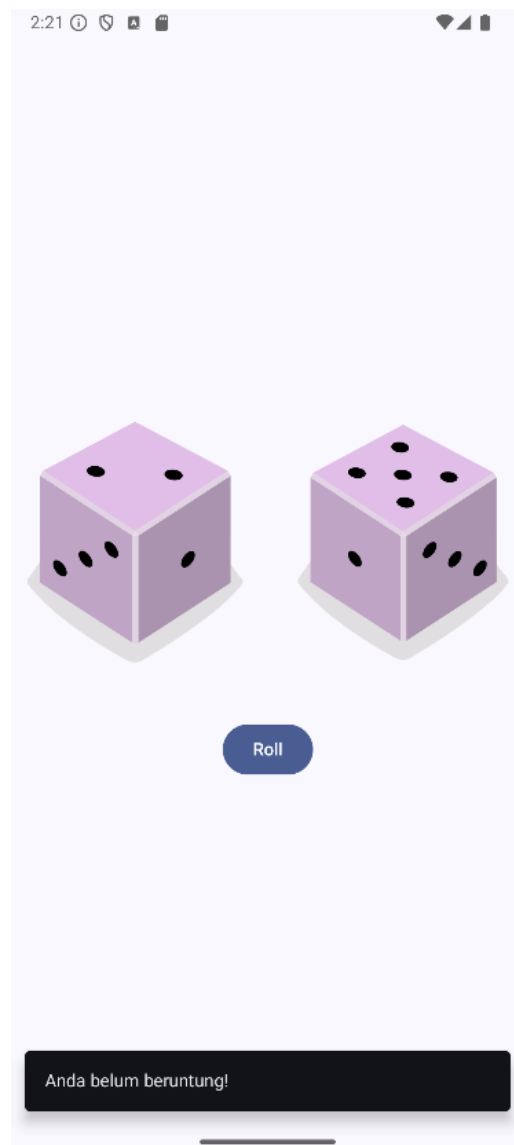
Gambar 5. Tampilan Dadu Setelah Di-Roll XML



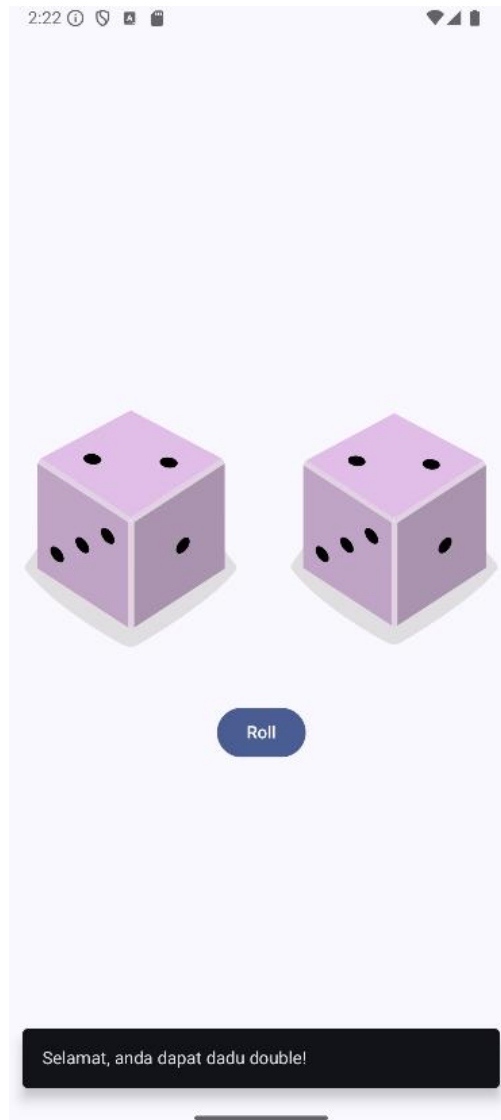
Gambar 6. Tampilan Roll Dadu Double XML



*Gambar 7. Tampilan Awal Aplikasi
Jetpack Compose*



*Gambar 8. Tampilan Dadu Setelah Di-Roll
Jetpack Compose*



Gambar 9. Tampilan Roll Dadu Double Jetpack Compose

C. Pembahasan

MainActivity.kt (XML):

1. Pada line [1], syntax `package com.example.diceroll` merupakan deklarasi nama package file Kotlin.
2. Pada line [3] sampai line [8], `import android.os.Bundle, import android.widget.Button, import android.widget.ImageView, import com.google.android.material.snackbar.Snackbar, import androidx.appcompat.app.AppCompatActivity, import com.example.diceroll2.R` merupakan program import class dan library yang diperlukan, seperti penyimpanan data sementara, mengatur komponen UI, menampilkan pesan pada layer, dan mengakses resource layout, drawable, dan lain-lain.
3. Pada line [10], syntax `class MainActivity : AppCompatActivity()` `{}` merupakan deklarasi kelas bernama MainActivity yang extend dari AppCompatActivity.
4. Pada line [12] sampai line [15], mereferensikan komponen UI dari layout XML, dan lateinit berarti variable akan diinisialisasi nanti.
5. Pada line [17] sampai line [25], deklarasi list variabel `diceImage`.
6. Pada line [27] sampai line [29], fungsi yang digunakan untuk menetapkan layout XML sebagai UI tampilan utama.
7. Pada line [31] sampai line [34], menghubungkan variable Kotlin dengan komponen XML menggunakan Id.
8. Pada line [36] sampai line [38], logika saat tombol ditekan program akan dijalankan dan menghasilkan angka acak dari 1 sampai 6 pada dua dadu.
9. Pada line [40] sampai line [41], deklarasi untuk mengubah gambar dadu sesuai angka hasil random.
10. Pada line [43] sampai line [47], logika untuk menentukan pesan, jika kedua dadu sama, maka akan menampilkan pesan tertentu, dan sebaliknya.
11. Pada line [49], syntax `Snackbar.make(coordinatorLayout, pesan, Snackbar.LENGTH_SHORT).show()` merupakan deklarasi program untuk menampilkan snackbar di bawah layer dengan pesan yang ditentukan.

activity_main.xml:

1. Pada line [1], syntax `<?xml version="1.0" encoding="utf-8"?>` sebagai deklarasi XML.
2. Pada line [2] sampai [7], merupakan root layout, coordinator layout. Dimana `CoordinatorLayout` merupakan layout fleksibel yang bisa bekerja sama dengan elemen-elemen seperti `Snackbar` dan lain-lain. Kemudian ada `id = "coordinatorlayout"` sebagai anchor untuk `Snackbar` di `MainActivity.kt`. terakhir `padding = "24dp"` sebagai pengasih jarak dari sisi layer.
3. Pada line [9] sampai line [13], syntax `<LinearLayout` digunakan untuk Menyusun elemen dari atas ke bawah dan `android:gravity = "center"` untuk membuat konten terpusat secara horizontal.
4. Pada line [15] sampai line [19], syntax `<LinearLayout` digunakan untuk menyusun dua buah dadu secara horizontal dan `marginBottom` agar dadu tidak menempel ke tombol.
5. Pada line [21] sampai line [26], syntax `<ImageView` digunakan untuk menampilkan gambar dadu awal (`dice_0`) dengan ukuran tetap `100dp`, dan `marginEnd` untuk mengasih jarak ke dadu kedua.
6. Pada line [28] sampai line [32], syntax `<ImageView` digunakan untuk menampilkan gambar dadu kedua dengan menampilkan gambar dadu awal (`dice_0`), namun tanpa `marginEnd`.
7. Pada line [35] sampai line [39], syntax `<Button` digunakan untuk menampilkan tombol yang akan melempar dua dadu ketika diklik yang dihubungkan dengan `rollButton` di file Kotlin.

MainActivity.kt (Jetpack Compose):

1. Pada line [1], syntax `package com.example.diceroll` merupakan deklarasi nama package file Kotlin.
2. Pada line [3] sampai line [34], merupakan program import class dan library yang diperlukan, seperti `compose` (layout, button, image, dan sebagainya), `coroutines` untuk `Snackbar`, dan lain-lain.
3. Pada line [36] sampai line [45], syntax `class MainActivity : ComponentActivity() {}` merupakan activity untuk Jetpack Compose yang berisi program UI penuh di layer, mengatur UI dengan `compose`, dan program untuk tema dan memanggil `composable` utama.
4. Pada line [47] sampai line [51], merupakan fungsi utama `composable` dan dapat dipreview di Android Studio dengan adanya `@Preview`.

5. Pada line [53] sampai line [58], merupakan fungsi utama UI yang berfungsi untuk menampilkan dua dadu dan satu tbutton roll.
6. Pada line [59] sampai line [60], merupakan program untuk menyimpan hasil dadu sehingga Compose tahu kapan UI harus di-recompose.
7. Pada line [61] sampai line [62], syntax `val snackbarHostState` merupakan variable yang digunakan untuk mengelola notifikasi (snackbar), kemudian `val coroutineScope` digunakan untuk menampilkan snackbar secara asynchronous.
8. Pada line [64] sampai line [81], merupakan program untuk menentukan gambar mana yang akan ditampilkan berdasarkan angka `result1` dan `result2`.
9. Pada line [84] sampai line [125], syntax `Scaffold()` merupakan kerangka UI Compose, `Row()` merupakan program untuk menaruh duan gambar dadu secara horizontal, `Spacer()` program untuk memberi jarak vertikal, dan `Button()` program untuk button yang Ketika diklik akan mengacak angka secara random dan menampilkan Snackbar dengan pesan.
10. Pada line [111] sampai line [138], merupakan fungsi button yang berisi program ketika button diklik untuk mengacak dua angka dadu dari 1-6, kemudian nilai akan disimpan ke dalam `result1` dan `result2` yang akan digunakan untuk menentukan gambar dadu. Kemudian, menentukan pesan yang sesuai dengan kondisi apakah angka berbeda atau sama (double) yang nilainya akan disipan ke dalam variable `message`. `coroutineScope.lunch` merupakan program untuk membuat coroutine baru yang bisa berjalan secara asynchronous tanpa menghambat UI. Di dalamnya terdapat program agar Ketika masih ada snackbar yang muncul sebelumnya akan ditutup terlebih dahulu agar tidak tumpang tindih, Ketika ada `dismissJob` sebelumnya yang aktif akan dibatalkan terlebih dahulu untuk mencegah banyak `delay()`, membuat coroutine baru dan menyimpannya ke dalam `dismissJob` yang akan memrogram waktu selama 2 detik sebelum lanjut ke baris berikutnya, dan akan ditutup secara otomatis.
11. Pada line [139], syntax `Text(stringResource(R.string.roll))` digunakan untuk menampilkan teks tombol dari `strings.xml`.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

<https://github.com/rizkiiirr/PraktikumMobile/tree/main/Modul%201>