LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 1

ANDROID BASIC WITH KOTLIN

Oleh:

Muhammad Adh-Dhiya'Us Salim NIM. 2310817210022

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MARET 2025

LEMBAR PENGESAHAN
LAPORAN PRAKTIKUM PEMROGRAMAN MOBILE
MODUL 1

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

Nama Praktikan : Muhammad Adh-Dhiya'Us Salim
NIM : 2310817210022

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Salsabila Syifa NIM. 2010817320004	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

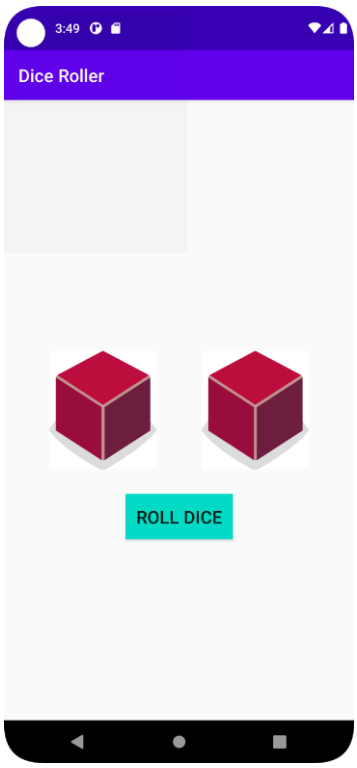
DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

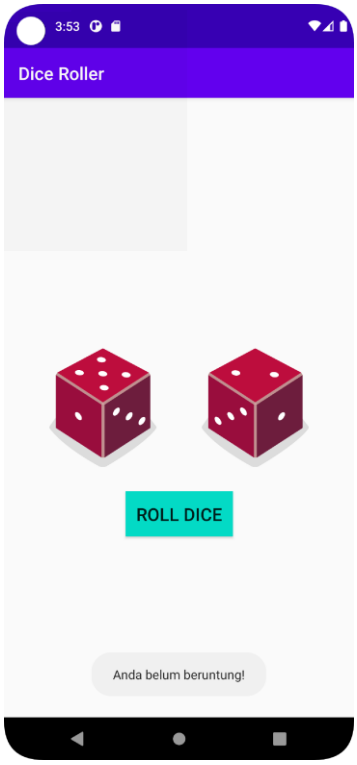
Soal Praktikum:
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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 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.

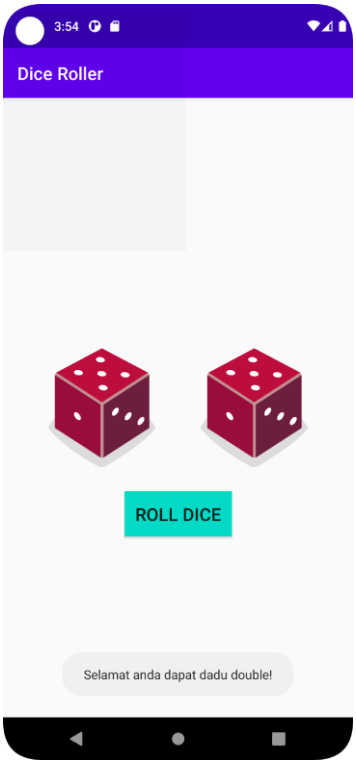


Gambar 2 Tampilan Dadu Setelah Di Roll

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

MainActivity.kt / Compose

```
1   package com.example.rolldadu
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.compose.foundation.Image
7   import androidx.compose.foundation.layout.*
8   import androidx.compose.material3.*
9   import androidx.compose.runtime.*
10  import androidx.compose.ui.Alignment
11  import androidx.compose.ui.Modifier
12  import androidx.compose.ui.res.painterResource
13  import androidx.compose.ui.unit.dp
14  import com.example.rolldadu.ui.theme.RollDaduTheme
15  import kotlin.random.Random
16  import androidx.compose.ui.graphics.Color
17  import kotlinx.coroutines.launch
18  import androidx.compose.material3.SnackbarHost
19  import androidx.compose.material3.SnackbarHostState
20  import androidx.compose.material3.Scaffold
21  import androidx.compose.material3.TopAppBar
22  import androidx.compose.material3.TopAppBarDefaults
```

```
23
24
25  class MainActivity : ComponentActivity() {
26      override fun onCreate(savedInstanceState: Bundle?) {
27          super.onCreate(savedInstanceState)
28          setContent {
29              RollDaduTheme {
30                  Surface(
31                      modifier = Modifier.fillMaxSize(),
32                      color = Color.Black
33                  ) {
34                      DiceRollerApp()
35                  }
36              }
37          }
38      }
39  }
40
41  @OptIn(ExperimentalMaterial3Api::class)
42  @Composable
43  fun DiceRollerApp() {
44      val snackbarHostState = remember { SnackbarHostState() }
45      val scope = rememberCoroutineScope()
46
47      var dice1 by remember { mutableStateOf(0) }
48      var dice2 by remember { mutableStateOf(0) }
49
50      val message = when {
51          dice1 == 0 && dice2 == 0 -> ""
52          dice1 == dice2 -> "Selamat, anda dapat dadu double!"
53          else -> "Anda belum beruntung!"
54      }
55
56      Scaffold(
57          modifier = Modifier.fillMaxSize(),
58          containerColor = Color.Black,
59          snackbarHost = {
60              SnackbarHost(hostState = snackbarHostState)
61          },
62          topBar = {
63              TopAppBar(
64                  title = {
65                      Text(
66                          text = "Dice Roller",
67                          color = Color.White
68                      )
69                  },
70                  colors = TopAppBarDefaults.topAppBarColors(
71                      containerColor = Color(0xFF6A1B9A)
72                  )
73              )
74          }
```

```
75      ) { padding ->
76          Column(
77              modifier = Modifier
78                  .fillMaxSize()
79                  .padding(padding)
80                  .padding(16.dp),
81              verticalArrangement = Arrangement.Center,
82              horizontalAlignment = Alignment.CenterHorizontally
83          ) {
84              Row(
85                  modifier = Modifier.fillMaxWidth(),
86                  horizontalArrangement =
87  Arrangement.SpaceEvenly
88              ) {
89                  DiceImage(dice1)
90                  DiceImage(dice2)
91              }
92
93              Spacer(modifier = Modifier.height(24.dp))
94
95              Button(
96                  onClick = {
97                      val newDice1 = Random.nextInt(1, 7)
98                      val newDice2 = Random.nextInt(1, 7)
99
100                     dice1 = newDice1
101                     dice2 = newDice2
102
103                     val resultMessage = if (newDice1 ==
104 newDice2) {
105                         "Selamat, anda dapat dadu double!"
106                     } else {
107                         "Anda belum beruntung!"
108                     }
109
110                     scope.launch {
112
113 snackbarHostState.currentSnackbarData?.dismiss()
114
115 snackbarHostState.showSnackbar(resultMessage)
116                     }
117                 },
118                 colors = ButtonDefaults.buttonColors(
119                     containerColor = Color(0xFF6A1B9A)
120                 )
121             ) {
123                 Text("Roll Dice", color = Color.White)
124             }
125         }
126     }
127 }
128
```

```
129
130 @Composable
131 fun            DiceImage(diceValue:            Int)            {
132     val       imageRes       =       when       (diceValue)       {
133         1                  ->                  R.drawable.dice_1
134         2                  ->                  R.drawable.dice_2
135         3                  ->                  R.drawable.dice_3
136         4                  ->                  R.drawable.dice_4
137         5                  ->                  R.drawable.dice_5
138         6                  ->                  R.drawable.dice_6
139         else                ->                 R.drawable.dice_0
140     }
141
142     Image(
143         painter    =    painterResource(id    =    imageRes),
144         contentDescription      =      "Dice      $diceValue",
145         modifier            =            Modifier.size(180.dp)
146     )
147 }
```

Table 1 Soruce Code Soal 1

activity_main.xml / XML

```
1  <?xml              version="1.0"              encoding="utf-8"?>
2  <LinearLayout
3  xmlns:android="http://schemas.android.com/apk/res/android"
4      android:id="@+id/rootLayout"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@android:color/black"
8      android:orientation="vertical"
9      android:gravity="center"
10     android:padding="24dp">
11
12     <LinearLayout
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:gravity="center"
16         android:orientation="horizontal"
17         android:layout_marginBottom="24dp">
18
19         <ImageView
20             android:id="@+id/diceImage1"
21             android:layout_width="140dp"
22             android:layout_height="140dp"
23             android:src="@drawable/dice_0"                    />
24
25         <Space
26             android:layout_width="16dp"
27             android:layout_height="match_parent"              />
28
```

```
        <ImageView
            android:id="@+id/diceImage2"
            android:layout_width="140dp"
            android:layout_height="140dp"
            android:src="@drawable/dice_0" />
    </LinearLayout>

    <Button
        android:id="@+id/rollButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Roll"
        android:backgroundTint="#6A1B9A"
        android:textColor="@android:color/white" />
</LinearLayout>
```

Table 2 Soruce Code Soal 1

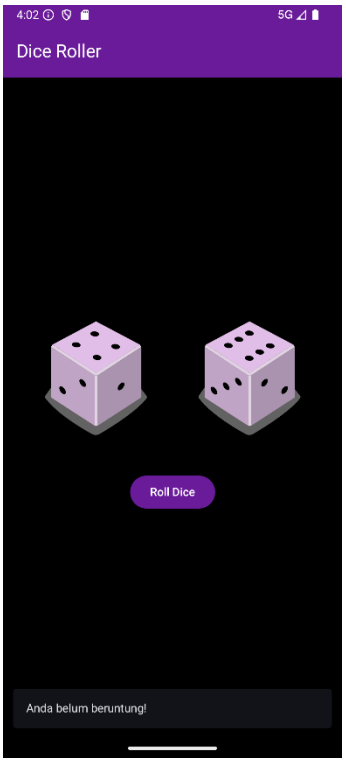
MainActivity.kt / XML

```
package com.example.rollxmldadu

import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.example.rollxmldadu.databinding.ActivityMainBinding
import kotlin.random.Random


class
MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    private val diceImages = listOf(
        R.drawable.dice_0,
        R.drawable.dice_1,
        R.drawable.dice_2,
        R.drawable.dice_3,
        R.drawable.dice_4,
        R.drawable.dice_5,
        R.drawable.dice_6
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.rollButton.setOnClickListener {
            val dice1 = Random.nextInt(6)
            val dice2 = Random.nextInt(6)

```

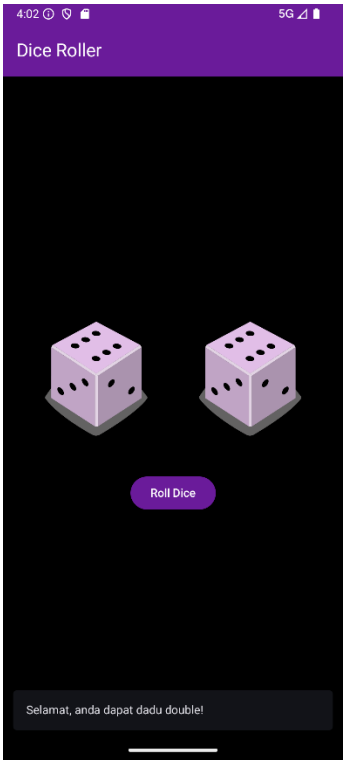
```
34
35  binding.diceImage1.setImageResource(diceImages[dice1])
36
37  binding.diceImage2.setImageResource(diceImages[dice2])
38
39              val   message   =   if   (dice1   ==   dice2)   {
40                  "Selamat,    anda    dapat    dadu    double!"
41              }                       else                    {
42                  "Anda             belum             beruntung!"
43              }
44
45              Toast.makeText(this,                        message,
46  Toast.LENGTH_SHORT).show()
47          }
48      }
49  }
```

Table 3 Soruce Code Soal 1

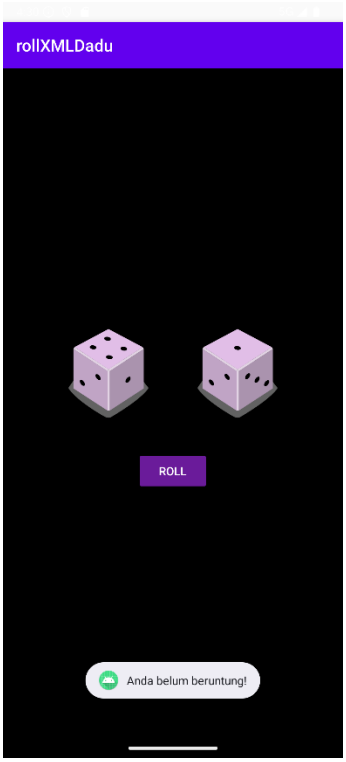
B. Output Program



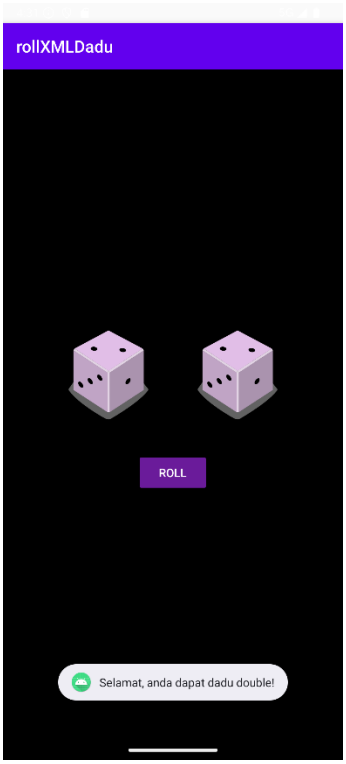
Gambar 1 Screenshot Hasil Jawaban Soal 1 - Compose



Gambar 2 Screenshot Hasil Jawaban Soal 1 - Compose



Gambar 3 Screenshot Hasil Jawaban Soal 1 - XML



Gambar 4 Screenshot Hasil Jawaban Soal 1 - XML

C. Pembahasan

MainActivity.kt / XML :

- **Pada baris 1**, dideklarasikan nama package untuk file Kotlin ini, yang berfungsi sebagai identitas unik lokasinya dalam struktur proyek.
- **Pada baris 3-7**, dilakukan proses import untuk beberapa pustaka yang dibutuhkan:
    - Bundle diimpor untuk menangani penyimpanan data atau state dari activity.
    - Toast diimpor untuk fungsionalitas menampilkan pesan notifikasi singkat di layar.
    - AppCompatActivity diimpor untuk digunakan sebagai kelas dasar (*base class*) sebuah activity agar mendukung fitur-fitur modern di berbagai versi Android.
    - ActivityMainBinding diimpor. Ini adalah kelas yang dibuat otomatis oleh fitur *View Binding* untuk mengakses komponen UI dari file activity_main.xml secara aman dan efisien.
    - Random diimpor dari pustaka standar Kotlin untuk fungsi pembuatan angka acak.
- **Pada baris 10**, dideklarasikan kelas utama bernama MainActivity yang merupakan turunan (*extends*) dari kelas AppCompatActivity.
- **Pada baris 12**, dideklarasikan variabel binding dengan tipe ActivityMainBinding. Variabel ini nantinya akan mereferensikan semua komponen UI pada layout. Penggunaan lateinit menandakan bahwa inisialisasi nilainya akan dilakukan nanti, bukan saat deklarasi.
- **Pada baris 14**, dideklarasikan sebuah List yang tidak bisa diubah (val) bernama diceImages. List ini berisi kumpulan referensi ID ke semua gambar dadu (dari dice_0 hingga dice_6) yang tersimpan di dalam folder res/drawable.
- **Pada baris 24**, dilakukan *override* pada fungsi onCreate, yang merupakan fungsi yang pertama kali dipanggil secara otomatis oleh sistem Android saat activity ini dibuat atau dimulai.
- **Pada baris 25**, dipanggil implementasi fungsi onCreate dari kelas induknya (super.onCreate) untuk memastikan semua konfigurasi esensial dari sebuah activity dijalankan.
- **Pada baris 26**, variabel binding yang sebelumnya dideklarasikan, kini diinisialisasi dengan cara "mengubah" atau me-*inflate* layout activity_main.xml menjadi sebuah objek binding yang dapat diakses oleh kode Kotlin.

- **Pada baris 27**, ditetapkan tampilan antarmuka (*user interface*) untuk activity ini dengan menggunakan layout akar (root) dari objek binding yang telah diinisialisasi.
- **Pada baris 29**, diberikan sebuah *event listener* pada komponen UI rollButton. Ini berarti blok kode yang ada di dalamnya akan dieksekusi setiap kali tombol tersebut diklik oleh pengguna.
- **Pada baris 30**, dideklarasikan variabel lokal dice1 dan langsung diisi dengan sebuah angka integer acak yang dihasilkan oleh Random.nextInt(6), yang akan memberikan nilai dari 0 hingga 5.
- **Pada baris 31**, dideklarasikan variabel lokal dice2 dan diisi dengan sebuah angka integer acak kedua, juga dengan rentang nilai dari 0 hingga 5.
- **Pada baris 33**, diatur sumber gambar (*image resource*) untuk komponen ImageView bernama diceImage1. Gambar yang dipilih diambil dari List diceImages pada indeks yang sesuai dengan nilai acak dari dice1.
- **Pada baris 34**, diatur sumber gambar untuk komponen ImageView diceImage2 berdasarkan nilai acak yang disimpan di variabel dice2.
- **Pada baris 36**, dideklarasikan variabel message yang nilainya akan ditentukan berdasarkan hasil dari sebuah ekspresi kondisional if-else.
- **Pada baris 37**, jika kondisi if pada baris sebelumnya (yaitu dice1 == dice2) terpenuhi, maka variabel message akan diisi dengan teks "Selamat, anda dapat dadu double!".
- **Pada baris 39**, jika kondisi if tidak terpenuhi (masuk ke dalam blok else), maka variabel message akan diisi dengan teks "Anda belum beruntung!".
- **Pada baris 42**, dibuat dan ditampilkan sebuah Toast (pesan pop-up singkat) di layar yang berisi teks yang telah disimpan di dalam variabel message.

activity_main.xml / XML :

1. Layout Utama (LinearLayout)

- **Pada baris 2**, didefinisikan LinearLayout sebagai elemen layout utama atau akar (*root*) dari tampilan ini.
- **Pada baris 3**, atribut android:id="@+id/rootLayout" memberikan identitas unik "rootLayout" pada LinearLayout ini.
- **Pada baris 4**, atribut android:layout_width="match_parent" mengatur agar lebar layout ini mengisi seluruh lebar layar perangkat.
- **Pada baris 5**, atribut android:layout_height="match_parent" mengatur agar tinggi layout ini mengisi seluruh tinggi layar perangkat.
- **Pada baris 6**, atribut android:background="@android:color/black" mengatur warna latar belakang layout menjadi hitam.

- **Pada baris 7**, atribut android:orientation="vertical" mengatur agar semua komponen anak (views) di dalamnya akan disusun secara vertikal dari atas ke bawah.
- **Pada baris 8**, atribut android:gravity="center" mengatur agar semua komponen anak di dalamnya diposisikan di tengah-tengah layout (secara horizontal dan vertikal).
- **Pada baris 9**, atribut android:padding="24dp" memberikan jarak (padding) sebesar 24dp di antara batas layout dengan konten di dalamnya.

2. Layout Penampung Dadu (LinearLayout)

- **Pada baris 11**, didefinisikan LinearLayout kedua yang berfungsi untuk menampung dua gambar dadu.
- **Pada baris 12**, lebarnya diatur match_parent agar mengisi lebar dari layout induknya.
- **Pada baris 13**, tingginya diatur wrap_content yang berarti tingginya akan menyesuaikan dengan tinggi konten di dalamnya (dalam hal ini, tinggi gambar dadu).
- **Pada baris 14**, android:gravity="center" digunakan untuk memposisikan komponen anak (dua gambar dadu) di tengah-tengah layout ini secara horizontal.
- **Pada baris 15**, android:orientation="horizontal" mengatur agar komponen di dalamnya disusun secara horizontal dari kiri ke kanan.
- **Pada baris 16**, android:layout_marginBottom="24dp" memberikan jarak bawah (margin) sebesar 24dp antara layout ini dengan komponen di bawahnya (yaitu tombol "Roll").

3. Komponen Gambar Dadu (ImageView dan Space)

- **Pada baris 18**, didefinisikan sebuah ImageView untuk menampilkan gambar dadu pertama.
- **Pada baris 19**, android:id="@+id/diceImage1" memberikan ID unik "diceImage1" agar bisa dimanipulasi dari kode Kotlin.
- **Pada baris 20 & 21**, android:layout_width dan android:layout_height mengatur ukuran gambar menjadi 140dp x 140dp.
- **Pada baris 22**, android:src="@drawable/dice_0" mengatur gambar awal (default) yang akan ditampilkan adalah dice_0.xml dari folder drawable.
- **Pada baris 24**, didefinisikan komponen Space yang berfungsi untuk memberikan jarak atau ruang kosong horizontal di antara dua gambar dadu. Lebarnya diatur 16dp.
- **Pada baris 29**, didefinisikan ImageView kedua dengan ID diceImage2, dengan konfigurasi ukuran dan gambar awal yang sama dengan ImageView pertama.

4. Tombol Aksi (Button)

- **Pada baris 37**, didefinisikan sebuah komponen Button.
- **Pada baris 38**, android:id="@+id/rollButton" memberikan ID unik "rollButton" untuk tombol ini, yang digunakan untuk mendeteksi event klik di kode Kotlin.
- **Pada baris 39 & 40**, android:layout_width dan android:layout_height diatur wrap_content, yang berarti ukuran tombol akan menyesuaikan dengan ukuran teks di dalamnya.
- **Pada baris 41**, android:text="Roll" mengatur teks yang akan ditampilkan pada tombol tersebut.
- **Pada baris 42**, android:backgroundTint="#6A1B9A" mengatur warna latar belakang tombol menjadi ungu.
- **Pada baris 43**, android:textColor="@android:color/white" mengatur warna teks pada tombol menjadi putih.

MainActivity.kt / Compose :

1. Kelas MainActivity

- **Pada baris 26**, kelas MainActivity dideklarasikan sebagai turunan dari ComponentActivity, yang merupakan kelas dasar untuk activity yang menggunakan Jetpack Compose.
- **Pada baris 28**, di dalam fungsi onCreate, dipanggil setContent. Ini adalah titik masuk utama untuk Jetpack Compose. Semua kode di dalam blok setContent berfungsi untuk mendefinisikan dan membangun antarmuka pengguna (UI).
- **Pada baris 30**, RollDaduTheme digunakan sebagai pembungkus UI. Ini adalah sebuah Composable yang menerapkan tema (seperti warna, bentuk, dan gaya teks) yang konsisten ke seluruh komponen di dalamnya.
- **Pada baris 31-34**, Surface digunakan sebagai container. Atribut modifier diatur untuk membuat Surface mengisi seluruh layar (fillMaxSize) dan color diatur menjadi hitam sebagai warna latar belakang.
- **Pada baris 35**, DiceRollerApp() dipanggil. Ini adalah fungsi Composable utama yang berisi semua logika dan komponen UI untuk aplikasi lempar dadu.

2. Fungsi Composable DiceRollerApp

Fungsi ini adalah inti dari aplikasi yang menampilkan semua elemen visual dan menangani logika.

- **Pada baris 41**, anotasi @OptIn(ExperimentalMaterial3Api::class) digunakan karena beberapa komponen Material 3 (seperti Scaffold dan TopAppBar) masih dianggap eksperimental dan memerlukan persetujuan eksplisit untuk digunakan.
- **Pada baris 42**, anotasi @Composable menandakan bahwa DiceRollerApp adalah sebuah fungsi Composable, yaitu blok bangunan dasar UI di Compose yang dapat memancarkan (emit) UI.
- **Pada baris 44**, val snackbarHostState = remember { SnackbarHostState() } digunakan untuk membuat dan mengingat (*remember*) state dari SnackbarHost, yang berfungsi untuk mengontrol kapan Snackbar (notifikasi di bagian bawah layar) harus muncul.
- **Pada baris 45**, val scope = rememberCoroutineScope() digunakan untuk mendapatkan sebuah *coroutine scope* yang terikat pada siklus hidup Composable ini. Ini akan digunakan untuk menjalankan operasi asinkron seperti menampilkan Snackbar.

- **Pada baris 47 & 48**, dideklarasikan dua variabel state, dice1 dan dice2. mutableStateOf(0) membuat sebuah state yang bisa diobservasi. remember memastikan state ini tidak direset setiap kali UI digambar ulang (*recomposition*). Setiap kali nilai dice1 atau dice2 berubah, semua Composable yang menggunakan state ini akan otomatis diperbarui.
- **Pada baris 50-54**, dideklarasikan variabel message menggunakan when untuk menentukan teks berdasarkan nilai dice1 dan dice2. *(Catatan: Variabel message ini sebenarnya tidak digunakan di dalam UI, karena logika serupa dideklarasikan ulang di dalam tombol 'Roll').*
- **Pada baris 56**, Scaffold digunakan. Ini adalah Composable yang menyediakan struktur layout Material Design standar, dengan slot untuk TopAppBar, SnackbarHost, dan konten utama.
- **Pada baris 59**, snackbarHost diatur untuk menampilkan Snackbar yang dikontrol oleh snackbarHostState.
- **Pada baris 62-72**, topBar diatur untuk menampilkan TopAppBar (bilah judul di bagian atas). Di dalamnya, judul, warna teks, dan warna latar belakang TopAppBar didefinisikan.
- **Pada baris 73**, blok padding dari Scaffold dimulai. padding ini berisi nilai padding yang harus diterapkan pada konten utama agar tidak tertutup oleh TopAppBar.
- **Pada baris 74**, Column digunakan untuk menyusun komponen di dalamnya secara vertikal. modifier digunakan untuk mengatur agar Column mengisi seluruh layar, menerapkan padding dari Scaffold, dan memposisikan semua kontennya di tengah.
- **Pada baris 82**, Row digunakan untuk menyusun gambar dadu secara horizontal. Arrangement.SpaceEvenly mendistribusikan ruang kosong secara merata di antara dan di sekitar gambar dadu.
- **Pada baris 86 & 87**, DiceImage(dice1) dan DiceImage(dice2) dipanggil. Ini adalah pemanggilan fungsi Composable DiceImage yang dapat digunakan kembali, dengan meneruskan nilai state dice1 dan dice2 sebagai parameter.
- **Pada baris 90**, Spacer digunakan untuk memberikan ruang kosong vertikal setinggi 24.dp antara barisan dadu dan tombol.
- **Pada baris 92**, Button Composable didefinisikan.
- **Pada baris 93**, di dalam lambda onClick, semua logika yang akan dijalankan saat tombol ditekan didefinisikan.
- **Pada baris 94 & 95**, dua angka acak baru dihasilkan menggunakan Random.nextInt(1, 7), yang akan menghasilkan nilai dari 1 hingga 6.

- **Pada baris 97 & 98**, nilai state dice1 dan dice2 diperbarui dengan angka acak yang baru. Pembaruan ini secara otomatis akan memicu *recomposition* pada Image dan UI lain yang bergantung pada state ini.
- **Pada baris 100-104**, logika pesan untuk Snackbar didefinisikan.
- **Pada baris 106**, scope.launch digunakan untuk memulai sebuah coroutine.
- **Pada baris 107 & 108**, snackbarHostState.showSnackbar(resultMessage) dipanggil untuk menampilkan Snackbar dengan pesan yang sesuai. Pemanggilan dismiss() sebelumnya memastikan Snackbar yang lama hilang sebelum yang baru muncul.
- **Pada baris 111-113**, colors pada Button diatur untuk mengubah warna latar belakang tombol.

3. Fungsi Composable DiceImage

Fungsi ini adalah komponen UI kecil yang dapat digunakan kembali, bertanggung jawab hanya untuk menampilkan satu gambar dadu.

- **Pada baris 119**, fungsi Composable DiceImage dideklarasikan, yang menerima satu parameter diceValue berupa Int.
- **Pada baris 120**, val imageRes dideklarasikan. Nilainya ditentukan oleh ekspresi when yang memetakan nilai diceValue (1-6) ke ID drawable yang sesuai. Jika nilainya di luar rentang (seperti pada kondisi awal yaitu 0), gambar dice_0 akan digunakan sebagai default.
- **Pada baris 129**, Image Composable digunakan untuk menampilkan gambar.
- **Pada baris 130**, painterResource(id = imageRes) digunakan untuk memuat gambar dari drawable.
- **Pada baris 131**, contentDescription diatur untuk tujuan aksesibilitas.
- **Pada baris 132**, modifier digunakan untuk mengatur ukuran Image menjadi 180dp x 180dp.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/Adiyaus/Praktikum-PemrogramanMobile.git