LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 1

ANDROID BASIC WITH KOTLIN

Oleh:

Muhammad Firas NIM. 2210817110014

PROGRAM STUDI TEKNOLOGI INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS LAMBUNG MANGKURAT
MARET 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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Muhammad Firas
NIM : 2210817110014

Menyetujui, Asisten Praktikum	Mengetahui, Dosen Penanggung Jawab Praktikum
Muhammad Raka Azwar NIM. 2210817210012	Andreyan Rizky Baskara, S.Kom., M.Kom. NIP. 19930703 201903 01 011

DAFTAR ISI

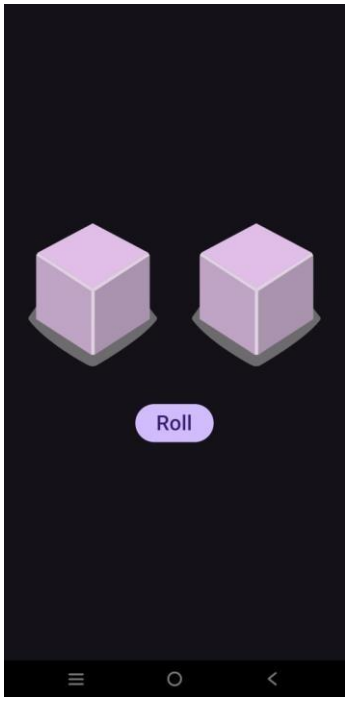
DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

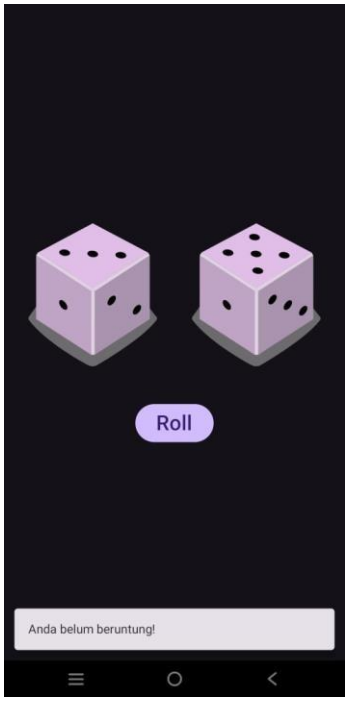
Soal Praktikum:
Buatlah sebuah aplikasi yang dapat menampilkan 2 buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



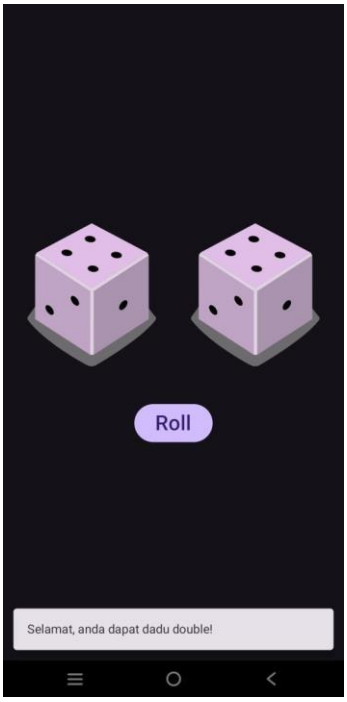
Gambar 1. Tampilan Awal Aplikasi

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



Gambar 2. Tampilan Dadu Setelah Di-Roll Gambar

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 yang 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 ke dalam repository GitHub ke dalam **folder Modul 1 dalam bentuk Project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repository.
6. Untuk gambar dadu dapat didownload pada link berikut: https://drive.google.com/file/d/14V3qXGdFnuoYN4AGd_9SgFh8kw8X9ySm/view?usp= sharing

A. Source Code

MainActivity.kt

Tabel 1. Source Code Jawaban Soal 1 MainActivity XML

```
1  package com.example.xmldice
2
3  import android.graphics.Color
4  import android.os.Bundle
5  import android.widget.Button
6  import android.widget.ImageView
7  import android.widget.TextView
8  import android.widget.Toast
9  import androidx.appcompat.app.AppCompatActivity
10 import kotlin.random.Random
11
12 class MainActivity : AppCompatActivity() {
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
```

```
15          setContentView(R.layout.activity_main)
16
17          val dice1: ImageView = findViewById(R.id.imageView1)
18          val dice2: ImageView = findViewById(R.id.imageView2)
19          val rollButton: Button = findViewById(R.id.button)
20          val messageText: TextView = findViewById(R.id.textView)
21
22          rollButton.setOnClickListener {
23              val dice1Value = Random.nextInt(1, 7)
24              val dice2Value = Random.nextInt(1, 7)
25
26              dice1.setImageResource(getDiceResource(dice1Value))
27              dice2.setImageResource(getDiceResource(dice2Value))
28
29              if (dice1Value == dice2Value) {
30                  messageText.text = "Selamat, anda dapat dadu
    double!"
31              } else {
32                  messageText.text = "Anda belum beruntung!"
33              }
34
35              messageText.visibility = TextView.VISIBLE
36          }
37      }
38
39      private fun getDiceResource(value: Int): Int {
40          return when (value) {
41              1 -> R.drawable.dice_1
42              2 -> R.drawable.dice_2
43              3 -> R.drawable.dice_3
44              4 -> R.drawable.dice_4
45              5 -> R.drawable.dice_5
46              6 -> R.drawable.dice_6
47              else -> R.drawable.dice_0
48          }
49      }
50  }
```

activity_main.xml

Tabel 2. Source Code Jawaban Soal 1 activity_main XML

```
1   <?xml version="1.0" encoding="utf-8"?>
2   <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       android:layout_width="match_parent"
5       android:layout_height="match_parent"
6       android:background="#121212"
7       android:gravity="center"
8       android:orientation="vertical"
```

```
9       android:padding="16dp"
10      android:weightSum="3">
11
12      <View
13          android:layout_width="match_parent"
14          android:layout_height="131dp"
15          android:layout_weight="1" />
16
17      <LinearLayout
18          android:layout_width="wrap_content"
19          android:layout_height="wrap_content"
20          android:gravity="center"
21          android:orientation="horizontal">
22
23          <ImageView
24              android:id="@+id/imageView1"
25              android:layout_width="wrap_content"
26              android:layout_height="wrap_content"
27              android:layout_weight="1"
28              app:srcCompat="@drawable/dice_0" />
29
30          <Space
31              android:layout_width="16dp"
32              android:layout_height="wrap_content" />
33
34          <ImageView
35              android:id="@+id/imageView2"
36              android:layout_width="wrap_content"
37              android:layout_height="wrap_content"
38              android:layout_weight="1"
39              app:srcCompat="@drawable/dice_0" />
40
41      </LinearLayout>
42
43      <View
44          android:layout_width="match_parent"
45          android:layout_height="20dp" />
46
47      <Button
48          android:id="@+id/button"
49          android:layout_width="108dp"
50          android:layout_height="wrap_content"
51          android:backgroundTint="#E0AAFF"
52          android:text="Roll"
53          android:textColor="@color/black"
54          android:textSize="24dp" />
55
56      <View
57          android:layout_width="match_parent"
58          android:layout_height="16dp"
59          android:layout_weight="2" />
60
```

```
61      <TextView
62          android:id="@+id/textView"
63          android:layout_width="379dp"
64          android:layout_height="wrap_content"
65          android:layout_marginLeft="20dp"
66          android:background="#FFFFFF"
67          android:baselineAligned="false"
68          android:padding="12dp"
69          android:text=""
70          android:textColor="#FF000000"
71          android:visibility="invisible"
72          android:textSize="16dp" />
73
74  </LinearLayout>
```

MainActivity.kt

Tabel 3. Source Code Jawaban Soal 1 MainActivity Jetpack Compose

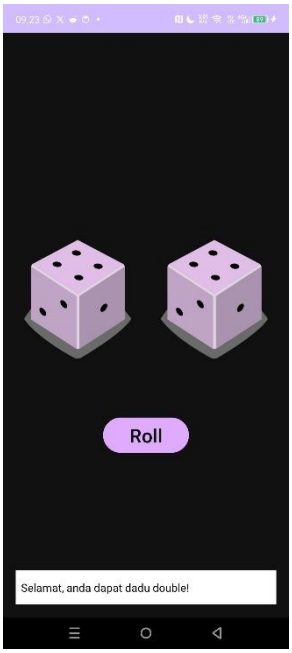
```
1   package com.example.jetpackcomposedice
2
3   import android.os.Bundle
4   import androidx.activity.ComponentActivity
5   import androidx.activity.compose.setContent
6   import androidx.activity.enableEdgeToEdge
7   import androidx.compose.foundation.Image
8   import androidx.compose.foundation.background
9   import androidx.compose.foundation.layout.*
10  import androidx.compose.foundation.shape.RoundedCornerShape
11  import androidx.compose.material3.*
12  import androidx.compose.runtime.*
13  import androidx.compose.runtime.saveable.rememberSaveable
14  import androidx.compose.ui.Alignment
15  import androidx.compose.ui.Modifier
16  import androidx.compose.ui.graphics.Color
17  import androidx.compose.ui.res.painterResource
18  import androidx.compose.ui.text.font.FontWeight
19  import androidx.compose.ui.tooling.preview.Preview
20  import androidx.compose.ui.unit.dp
21  import androidx.compose.ui.unit.sp
    import
    com.example.jetpackcomposedice.ui.theme.JetpackComposeDiceThem
22  e
23  import kotlin.random.Random
24
25  class MainActivity : ComponentActivity() {
26      override fun onCreate(savedInstanceState: Bundle?) {
27          super.onCreate(savedInstanceState)
28          enableEdgeToEdge()
29          setContent {
30              JetpackComposeDiceTheme {
                    Scaffold(modifier = Modifier.fillMaxSize()) {
```

```
31  innerPadding ->
32                      DiceScreen(modifier =
    Modifier.padding(innerPadding))
33                  }
34              }
35          }
36      }
37  }
38
39  @Composable
40  fun DiceImage(diceValue: Int) {
41      Image(
42          painter = painterResource(id =
    getDiceResource(diceValue)),
43          contentDescription = "Dice $diceValue",
44          modifier = Modifier.size(180.dp)
45      )
46  }
47
48  fun getDiceResource(value: Int): Int {
49      return when (value) {
50          1 -> R.drawable.dice_1
51          2 -> R.drawable.dice_2
52          3 -> R.drawable.dice_3
53          4 -> R.drawable.dice_4
54          5 -> R.drawable.dice_5
55          6 -> R.drawable.dice_6
56          else -> R.drawable.dice_0
57      }
58  }
59
60  @Composable
61  fun DiceScreen(modifier: Modifier = Modifier) {
62      var dice1 by rememberSaveable { mutableStateOf(0) }
63      var dice2 by rememberSaveable { mutableStateOf(0) }
64      var message by rememberSaveable { mutableStateOf("") }
65      var isRolled by rememberSaveable { mutableStateOf(false) }
66
67      Column(
68          modifier = modifier
69              .fillMaxSize()
70              .background(Color.Black)
71              .padding(0.dp),
72          verticalArrangement = Arrangement.SpaceBetween,
73          horizontalAlignment = Alignment.CenterHorizontally
74      ) {
75          Spacer(modifier = Modifier.weight(1f))
76
77          Row(
78              modifier = Modifier
79                  .fillMaxWidth(),
80              horizontalArrangement = Arrangement.Center
```

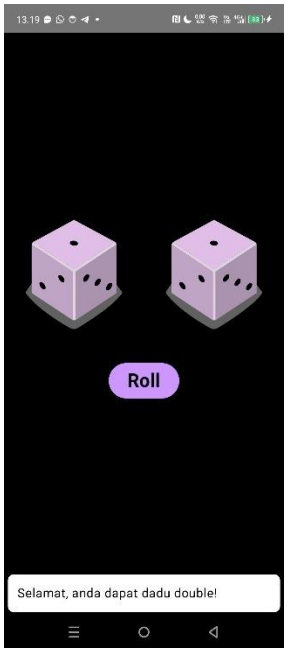
```
81          ) {
82              DiceImage(dice1)
83              Spacer(modifier = Modifier.width(0.dp))
84              DiceImage(dice2)
85          }
86
87          Spacer(modifier = Modifier.height(20.dp))
88
89          Button(
90              onClick = {
91                  dice1 = Random.nextInt(1, 7)
92                  dice2 = Random.nextInt(1, 7)
93                  isRolled = true
94                  message = if (dice1 == dice2) {
95                      "Selamat, anda dapat dadu double!"
96                  } else {
97                      "Anda belum beruntung!"
98                  }
99              },
100             colors =
    ButtonDefaults.buttonColors(containerColor =
101 Color(0xFFCE96FF))
102         ) {
103             Text(
104                 text = "Roll",
105                 fontSize = 24.sp,
106                 color = Color.Black,
107                 fontWeight = FontWeight.Bold)
108         }
109
110         Spacer(modifier = Modifier.weight(1f))
111
112         if (isRolled) {
113             Surface(
114                 color = Color.White,
115                 shape = RoundedCornerShape(8.dp),
116                 modifier = Modifier
117                     .fillMaxWidth()
118                     .padding(5.dp)
119             ) {
120                 Text(
121                     text = message,
122                     fontSize = 16.sp,
123                     color = Color.Black,
124                     modifier = Modifier.padding(12.dp)
125                 )
126             }
127         }
128     }
129 }
130
131 @Preview(showBackground = true)
```

```
@Composable
fun DicePreview() {
    JetpackComposeDiceTheme {
        DiceScreen()
    }
}
```

B. Output Program



Gambar 4. Screenshot Hasil Jawaban Soal 1 XML



Gambar 5. Screenshot Hasil Jawaban Soal Jetpack Compose

C. Pembahasan

MainActivity.kt (XML):

Pada baris [1], `package com.example.xmldice` berfungsi untuk mendeklarasikan bahwa file ini berada dalam package `com.example.xmldice`, yang merupakan bagian dari struktur proyek Android. Pada baris [3], `import android.graphics.Color` berfungsi untuk mengimpor kelas `Color` yang digunakan untuk mengatur warna dalam aplikasi. Pada baris [4], `import android.os.Bundle` berfungsi untuk mengimpor kelas `Bundle`, yang digunakan untuk menyimpan dan mengambil data saat aktivitas diinisialisasi. Pada baris [5], `import android.widget.Button` berfungsi untuk mengimpor kelas `Button`, yang digunakan untuk menampilkan dan mengelola tombol dalam UI aplikasi. Pada baris [6], `import android.widget.ImageView` berfungsi untuk mengimpor kelas `ImageView`, yang digunakan untuk menampilkan gambar dalam aplikasi. Pada baris [7], `import android.widget.TextView` berfungsi untuk mengimpor kelas `TextView`, yang digunakan untuk menampilkan teks di layar. Pada baris [8], `import android.widget.Toast` berfungsi untuk mengimpor kelas `Toast`, yang digunakan untuk menampilkan pesan singkat kepada pengguna. Pada baris [9], `import androidx.appcompat.app.AppCompatActivity` berfungsi untuk mengimpor kelas `AppCompatActivity`, yang merupakan kelas dasar untuk aktivitas dalam aplikasi Android modern. Pada baris [10], `import kotlin.random.Random` berfungsi untuk mengimpor `Random`, yang digunakan untuk menghasilkan angka acak.

Pada baris [12], `class MainActivity : AppCompatActivity() {` berfungsi untuk mendeklarasikan kelas `MainActivity`, yang merupakan aktivitas utama dalam aplikasi dan mewarisi `AppCompatActivity`. Pada baris [13], `override fun onCreate(savedInstanceState: Bundle?) {` berfungsi untuk meng-override metode `onCreate`, yang dipanggil saat aktivitas pertama kali dibuat. Pada baris [14], `super.onCreate(savedInstanceState)` berfungsi untuk memanggil implementasi `onCreate` dari `AppCompatActivity`, memastikan bahwa konfigurasi awal berjalan dengan baik. Pada baris [15], `setContentView(R.layout.activity_main)` berfungsi untuk mengatur tata letak tampilan aktivitas dengan menggunakan file XML `activity_main.xml`.

Pada baris [17], `val dice1: ImageView = findViewById(R.id.imageView1)` berfungsi untuk menghubungkan variabel `dice1` dengan elemen `ImageView` yang memiliki ID `imageView1` dalam layout XML. Pada baris [18], `val dice2: ImageView = findViewById(R.id.imageView2)` berfungsi untuk menghubungkan variabel `dice2` dengan elemen `ImageView` yang memiliki ID `imageView2`. Pada baris [19], `val rollButton: Button = findViewById(R.id.button)` berfungsi untuk menghubungkan variabel `rollButton` dengan elemen `Button` yang memiliki ID `button`. Pada baris [20], `val messageText: TextView = findViewById(R.id.textView)` berfungsi untuk menghubungkan variabel `messageText` dengan elemen `TextView` yang memiliki ID `textView`.

Pada baris [22], `rollButton.setOnClickListener {` berfungsi untuk menetapkan aksi yang akan dijalankan saat tombol ditekan. Pada baris [23], `val dice1Value = Random.nextInt(1, 7)` berfungsi untuk menghasilkan angka acak antara `1` hingga `6` untuk dadu pertama. Pada baris [24], `val dice2Value = Random.nextInt(1, 7)` berfungsi untuk menghasilkan angka acak antara `1` hingga `6` untuk dadu kedua. Pada baris [26], `dice1.setImageResource(getDiceResource(dice1Value))` berfungsi untuk mengubah gambar `ImageView` pertama sesuai dengan hasil dadu pertama. Pada baris [27], `dice2.setImageResource(getDiceResource(dice2Value))` berfungsi untuk mengubah gambar `ImageView` kedua sesuai dengan hasil dadu kedua.

Pada baris [29], `if (dice1Value == dice2Value) {` berfungsi untuk memeriksa apakah kedua dadu memiliki nilai yang sama. Pada baris [30], `messageText.text = "Selamat, anda dapat dadu double!"` berfungsi untuk mengubah teks dalam `TextView` menjadi "Selamat, anda dapat dadu double!" jika kedua dadu memiliki angka yang sama. Pada baris [31], `} else {` menandakan awal dari kondisi `else` yang akan dieksekusi jika nilai kedua dadu tidak sama. Pada baris [32], `messageText.text = "Anda belum beruntung!"` berfungsi untuk mengubah teks dalam `TextView` menjadi "Anda belum beruntung!" jika hasil dadu berbeda. Pada baris [35], `messageText.visibility = TextView.VISIBLE` berfungsi untuk memastikan bahwa `TextView` ditampilkan setelah tombol ditekan.

Pada baris [37], `}` menutup blok kode `setOnClickListener`. Pada baris [39], `private fun getDiceResource(value: Int): Int {` berfungsi untuk mendeklarasikan fungsi `getDiceResource`, yang mengembalikan sumber daya gambar dadu berdasarkan nilai yang diberikan. Pada baris [40-46], `return when (value) { ... }` adalah struktur `when` yang menentukan gambar yang sesuai dengan nilai dadu (`1-6`) menggunakan `R.drawable.dice_1` hingga `R.drawable.dice_6`. Pada baris [47], `else -> R.drawable.dice_0` digunakan untuk mengembalikan gambar default jika nilai tidak sesuai dengan angka dadu (`1-6`). Pada baris [49], `}` menutup fungsi `getDiceResource`. Pada baris [50], `}` menutup kelas `MainActivity`.

activity_main.xml (XML):

Pada baris [1], `<?xml version="1.0" encoding="utf-8"?>` digunakan untuk menyatakan bahwa dokumen ini adalah file XML dan menggunakan encoding UTF-8. Pada baris [2-10], `<LinearLayout>` digunakan sebagai root layout atau elemen utama dalam tampilan antarmuka, dengan orientasi `vertical`, yang artinya semua elemen di dalamnya akan disusun dari atas ke bawah. Atribut `xmlns:android` dan `xmlns:app` pada baris ini mendefinisikan namespace standar Android dan app-compat untuk atribut khusus. Atribut `android:layout_width="match_parent"` dan `android:layout_height="match_parent"` menandakan bahwa LinearLayout akan memenuhi seluruh lebar dan tinggi layar. Atribut `android:background="#121212"` memberi latar belakang berwarna hitam keabu-abuan. Atribut `android:gravity="center"` akan memusatkan isi layout. Atribut `android:padding="16dp"` memberi ruang di semua sisi. Atribut `android:weightSum="3"` digunakan untuk mengatur total pembagian bobot (weight) dari elemen-elemen yang menggunakannya.

Pada baris [12-15], tag `<View>` digunakan sebagai ruang kosong vertikal dengan tinggi 131dp dan bobot 1, untuk memberi jarak pada bagian atas layout.

Pada baris [17-21], `<LinearLayout>` baru dibuat di dalam layout utama, dengan orientasi `horizontal` agar elemen-elemen di dalamnya tersusun secara mendatar. Atribut `layout_width="wrap_content"` dan `layout_height="wrap_content"` menyesuaikan ukuran dengan kontennya, dan `gravity="center"` memusatkan elemen-elemen di tengah.

Pada baris [23-28], `<ImageView>` pertama digunakan untuk menampilkan gambar dadu. Atribut `android:id="@+id/imageView1"` memberi identitas unik untuk mengakses elemen ini di kode Kotlin. `layout_width` dan `layout_height` disesuaikan dengan ukuran gambar. `layout_weight="1"` membantu dalam pembagian ruang horizontal secara proporsional. Atribut `app:srcCompat="@drawable/dice_0"` menampilkan gambar dadu kosong (dice_0) dari resource drawable.

Pada baris [30-32], `<Space>` digunakan untuk memberi jarak horizontal antara dua gambar dadu, dengan lebar 16dp.

Pada baris [34-39], `<ImageView>` kedua mirip dengan yang pertama, hanya id-nya berbeda yaitu `@+id/imageView2`. Gambar default-nya juga dice_0.

Pada baris [41], tag `</LinearLayout>` digunakan untuk menutup layout horizontal tempat kedua dadu berada.

Pada baris [43-45], tag `<View>` kembali digunakan untuk memberi jarak antara bagian dadu dan tombol, dengan tinggi 20dp.

Pada baris [47-54], `<Button>` digunakan untuk menampilkan tombol "Roll". Atribut `android:id="@+id/button"` memberikan id agar dapat diakses di kode Kotlin. `layout_width="108dp"` menentukan lebar tombol. Atribut `backgroundTint="#E0AAFF"` memberikan warna ungu pastel sebagai latar tombol. Atribut `android:text="Roll"` menampilkan teks "Roll" di tombol tersebut. Atribut `android:textColor="@color/black"` menjadikan warna teks menjadi hitam. Atribut `textSize="24dp"` memberi ukuran font yang besar.

Pada baris [56-59], `<View>` digunakan lagi untuk memberi ruang kosong vertikal dengan `layout_weight="2"`, agar bagian bawah terasa lebih luas.

Pada baris [61-72], `<TextView>` digunakan untuk menampilkan hasil pelemparan dadu. Atribut `android:id="@+id/textView"` memberikan id agar dapat dimanipulasi dari kode Kotlin. `layout_width="379dp"` menentukan lebar teks, dan `layout_height="wrap_content"` menyesuaikan tinggi. Atribut `layout_marginLeft="20dp"` memberi jarak dari sisi kiri. Atribut `background="#FFFFFF"` memberi latar putih. Atribut `baselineAligned="false"` digunakan untuk pengaturan tampilan teks. Atribut `padding="12dp"` memberi ruang di dalam elemen. Atribut `text=""` artinya teks awalnya kosong. `textColor="#FF000000"` memberi warna hitam solid. Atribut

`visibility="invisible"` membuat TextView tidak terlihat saat aplikasi pertama kali dijalankan. `textSize="16dp"` mengatur ukuran teks.

Pada baris [63], `</LinearLayout>` digunakan untuk menutup tag LinearLayout utama dan mengakhiri dokumen XML tampilan.

MainActivity.kt (Jetpack Compose) :

Pada baris [1], `package com.example.jetpackcomposedice` mendefinisikan package tempat file ini berada, yaitu `com.example.jetpackcomposedice`, yang membantu mengorganisir kode dalam struktur proyek Android. Pada baris [3–23], berbagai `import` digunakan untuk membawa fungsi dan komponen dari Jetpack Compose, Android, dan Kotlin agar dapat digunakan di file ini, seperti `Image`, `Column`, `Modifier`, dan lainnya.

Pada baris [25], dideklarasikan `MainActivity` sebagai subclass dari `ComponentActivity`, yang merupakan entry point utama dari aplikasi Android berbasis Compose. Pada baris [26–32], override fungsi `onCreate()` digunakan untuk menginisialisasi aktivitas. Pada baris [27], `super.onCreate(savedInstanceState)` memanggil implementasi `onCreate` dari superclass. Pada baris [28], `enableEdgeToEdge()` digunakan untuk memungkinkan tampilan aplikasi mengisi seluruh layar. Pada baris [29–32], `setContent` digunakan untuk menampilkan UI menggunakan Jetpack Compose. Tema aplikasi `JetpackComposeDiceTheme` dibungkus di dalam `Scaffold`, dan komponen utama `DiceScreen` ditampilkan dengan padding bawaan dari `Scaffold`.

Pada baris [39], dideklarasikan fungsi `DiceImage` sebagai fungsi `@Composable`, yang berarti fungsi ini dapat membangun UI. Pada baris [40–44], komponen `Image` digunakan untuk menampilkan gambar dadu berdasarkan nilai `diceValue` yang diberikan. Nilai gambarnya diambil menggunakan fungsi `getDiceResource()`. Disini juga diberikan `Modifier` sebesar 180 dp untuk ukuran gambarnya

Pada baris [48–56], fungsi `getDiceResource()` dideklarasikan untuk mengembalikan resource ID dari gambar dadu berdasarkan nilai angka (1–6). Jika nilainya di luar rentang tersebut, gambar default `dice_0` akan ditampilkan.

Pada baris [60-61], fungsi `DiceScreen()` didefinisikan sebagai fungsi `@Composable`, yang berisi seluruh layout utama dari aplikasi. Parameter `modifier` digunakan untuk menambahkan styling atau behavior ke layout-nya. Pada baris [62–65], dibuat beberapa variabel `mutableStateOf` yang dibungkus dengan `rememberSaveable`, yaitu `dice1`, `dice2`, `message`, dan `isRolled`, untuk menyimpan state UI secara reaktif dan tahan terhadap rotasi layar.

Pada baris [67–73], komponen `Column` digunakan sebagai layout vertikal. `modifier` digunakan untuk mengisi seluruh layar, memberikan latar belakang hitam, dan padding 0. `Arrangement.SpaceBetween` digunakan untuk menyebar isi secara vertikal, sementara `Alignment.CenterHorizontally` memposisikan isi di tengah secara horizontal.

Pada baris [75], `Spacer` digunakan dengan `weight(1f)` agar memberi ruang fleksibel di bagian atas layar.

Pada baris [77–84], `Row` digunakan untuk menampilkan dua gambar dadu secara horizontal di tengah layar. Dua komponen `DiceImage` ditampilkan untuk `dice1` dan `dice2`, dengan `Spacer` di antaranya untuk memberi jarak (meskipun lebarnya 0dp).

Pada baris [87], ditambahkan `Spacer` vertikal dengan tinggi 20dp untuk memberi jarak antara dadu dan tombol. Pada baris [89–100], sebuah `Button` ditampilkan, dan ketika ditekan, akan meng-generate angka acak untuk `dice1` dan `dice2` (antara 1–6). Jika keduanya sama, akan muncul pesan "Selamat, anda dapat dadu double!" dan jika berbeda, muncul "Anda belum beruntung!". Warna tombol diatur dengan `ButtonDefaults.buttonColors`.

Pada baris [102-106], `Text` yang ditampilkan di button akan menampilkan tulisan "Roll", memiliki ukuran font 24 sp atau Scale Independent Pixels, memiliki warna `Text` Hitam, dan `fontWeight.Bold` untuk menebalkan tulisan.

Pada baris [109], `Spacer` lain digunakan di bagian bawah untuk memberi ruang fleksibel.

Pada baris [111], sebuah `if` statement digunakan untuk menampilkan pesan hanya jika `isRolled` bernilai `true`. Pada baris [112–117], `Surface` digunakan sebagai wadah berwarna putih dengan sudut membulat (`RoundedCornerShape`) dengan ukuran 8 dp atau Density Independent Pixels, kemudian digunakan `Modifier` untuk mengisi maksimal lebar layar dengan `padding` ukuran 5 dp.

Pada baris [119-123], komponen `Text` akan menampilkan `message` dari hasil lemparan dadu. Pesan ini diberi padding 12 dp, warna teks hitam, serta ukuran huruf 16 sp.

Pada baris [130-134], fungsi `DicePreview()` didefinisikan dengan anotasi `@Preview`, yang memungkinkan tampilan preview di Android Studio. Fungsi ini hanya memanggil `DiceScreen()` dalam tema `JetpackComposeDiceTheme`.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.
https://github.com/MuhammadFiras/Praktikum-Mobile/tree/main