LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 1

ANDROID BASIC WITH KOTLIN

Oleh:

Zulfa Auliya Akbar NIM. 2210817210026

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 Prakitkum ini dikerjakan oleh:

Nama Praktikan : Muhammad Fauzan Ahsani
NIM : 2310817310009

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

DAFTAR ISI

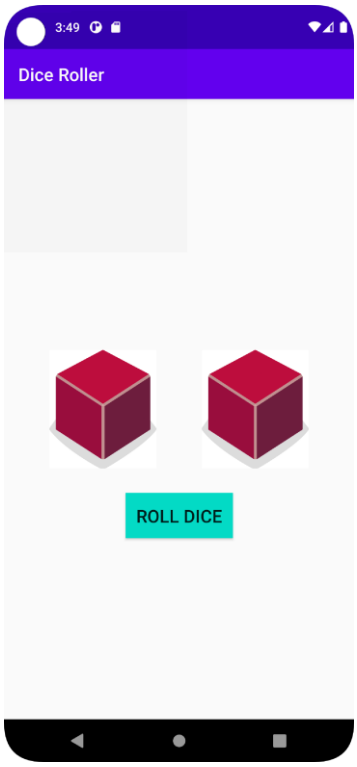
DAFTAR GAMBAR

DAFTAR TABEL

SOAL 1

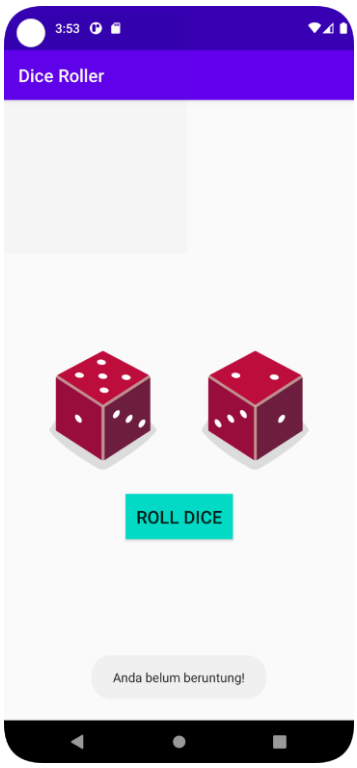
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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



Gambar 1. Tampilan Awal saat Aplikasi Dibuka

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.

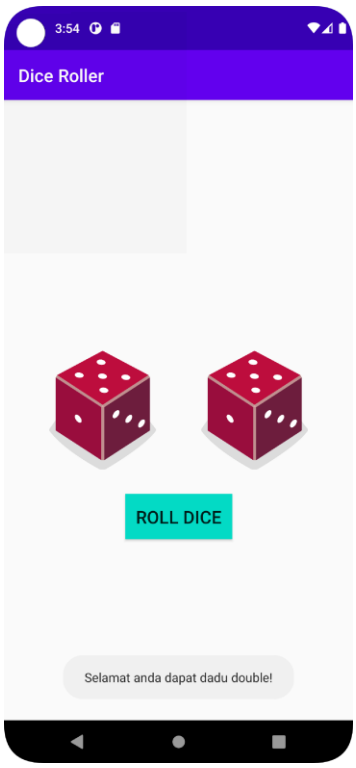


Gambar 2. Tampilan saat Dadu yang Diacak Hasilnya Berbeda

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

4. Upload aplikasi yang telah anda buat kedalam repository github ke dalam **folder Module 1 dalam bentuk project.** Jangan lupa untuk melakukan **Clean Project** sebelum mengupload pekerjaan anda pada repo.

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



Gambar 3. Tampilan saat Dadu yang Diacak Sama Nilainya

A. Source Code

1. MainActivity.kt

Tabel 1. Source Code MainActiviy.kt Jawaban Soal 1

```
 1  package com.example.dicerollercompose
 2
 3  import android.os.Bundle
 4  import android.view.Gravity
 5  import android.widget.Toast
 6  import androidx.activity.ComponentActivity
 7  import androidx.activity.compose.setContent
 8  import androidx.compose.foundation.Image
 9  import androidx.compose.foundation.layout.fillMaxSize
10  import androidx.compose.foundation.layout.wrapContentSize
11  import androidx.compose.runtime.Composable
12  import androidx.compose.ui.Alignment
13  import androidx.compose.ui.Modifier
14  import androidx.compose.foundation.layout.Column
15  import androidx.compose.foundation.layout.Row
16  import androidx.compose.material3.Button
17  import androidx.compose.material3.Text
18  import androidx.compose.ui.res.stringResource
19  import androidx.compose.ui.tooling.preview.Preview
20  import androidx.compose.foundation.layout.Spacer
21  import androidx.compose.foundation.layout.height
22  import androidx.compose.material3.MaterialTheme
```

```
23 import androidx.compose.material3.Surface
24 import androidx.compose.ui.res.painterResource
25 import androidx.compose.ui.unit.dp
26 import
   com.example.dicerollercompose.ui.theme.DiceRollerComposeTheme
27 import androidx.compose.ui.platform.LocalContext
28 import androidx.compose.ui.unit.sp
29 import androidx.lifecycle.viewmodel.compose.viewModel
30
31 class MainActivity : ComponentActivity() {
32
33     override fun onCreate(savedInstanceState: Bundle?) {
34         super.onCreate(savedInstanceState)
35         setContent {
36             DiceRollerComposeTheme {
37                 Surface(
38                     modifier = Modifier.fillMaxSize(),
39                     color = MaterialTheme.colorScheme.background
40                 ) {
41                     DiceRollerApp()
42                 }
43             }
44         }
45     }
46 }
47
48 @Preview
49 @Composable
50 fun DiceRollerApp(){
51     DiceWithButtonAndImage(modifier= Modifier
52         .fillMaxSize()
53         .wrapContentSize(Alignment.Center)
54     )
55 }
56
57 @Composable
58 fun DiceWithButtonAndImage(modifier: Modifier = Modifier){
59     val viewModel: DiceViewModel = viewModel()
60     val context = LocalContext.current
61
62     val imgDiceLeft = when (viewModel.numDiceLeft) {
63         1 -> R.drawable.dice_1
64         2 -> R.drawable.dice_2
65         3 -> R.drawable.dice_3
66         4 -> R.drawable.dice_4
67         5 -> R.drawable.dice_5
68         6 -> R.drawable.dice_6
69         else -> R.drawable.dice_0
70     }
71
72     val imgDiceRight = when (viewModel.numDiceRight) {
```

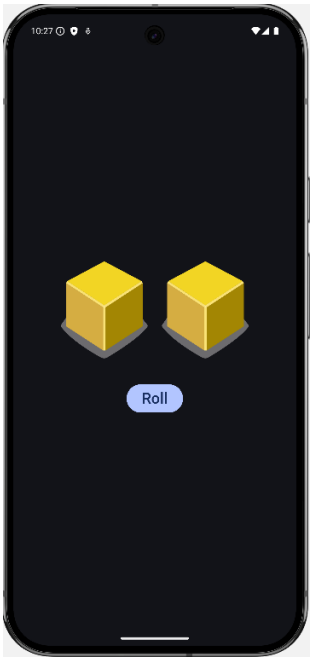
```
 73          1 -> R.drawable.dice_1
 74          2 -> R.drawable.dice_2
 75          3 -> R.drawable.dice_3
 76          4 -> R.drawable.dice_4
 77          5 -> R.drawable.dice_5
 78          6 -> R.drawable.dice_6
 79          else -> R.drawable.dice_0
 80      }
 81
 82      Column(
 83          modifier = modifier,
 84          horizontalAlignment = Alignment.CenterHorizontally
 85      ){
 86          Row {
 87              Image(
 88                  painter = painterResource(imgDiceLeft),
 89                  contentDescription =
    viewModel.numDiceLeft.toString(),
 90                  modifier = Modifier.height(200.dp)
 91              )
 92              Spacer(modifier = Modifier.height(100.dp))
 93              Image(
 94                  painter = painterResource(imgDiceRight),
 95                  contentDescription =
    viewModel.numDiceRight.toString(),
 96                  modifier = Modifier.height(200.dp)
 97              )
 98          }
 99          Spacer(modifier = Modifier.height(16.dp))
100
101          Button(
102              onClick = {
103                  viewModel.rollDice()
104
105                  val msg = viewModel.getResultMessage()
106
107                  val toast = Toast.makeText(context, msg,
    Toast.LENGTH_SHORT)
108                  toast.setGravity(Gravity.BOTTOM or
    Gravity.CENTER_HORIZONTAL, 0, 150)
109                  toast.show()
110              }) {
111              Text(stringResource(R.string.roll), fontSize =
    24.sp)
112          }
113      }
114  }
```

2. DiceViewModel.kt

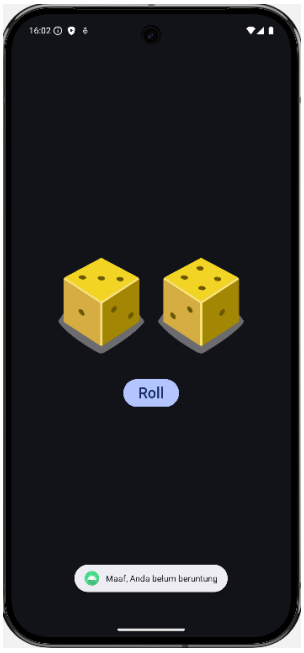
Tabel 2. Source Code DiceViewModel.kt Jawaban Soal 1

```
 1 package com.example.dicerollercompose
 2
 3 import androidx.compose.runtime.getValue
 4 import androidx.lifecycle.ViewModel
 5 import androidx.compose.runtime.mutableStateOf
 6 import androidx.compose.runtime.setValue
 7
 8 class DiceViewModel: ViewModel(){
 9     var numDiceLeft by mutableStateOf(0)
10         private set
11
12     var numDiceRight by mutableStateOf(0)
13         private set
14
15     fun rollDice() {
16         numDiceLeft = (1..6).random()
17         numDiceRight = (1..6).random()
18     }
19
20     fun getResultMessage(): String {
21         return if (numDiceLeft == numDiceRight)
22             "Selamat, Anda mendapatkan nilai double"
23         else
24             "Maaf, Anda belum beruntung"
25     }
26 }
```

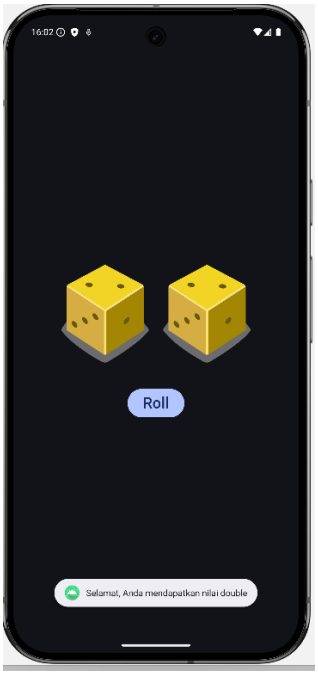
B. Output Program



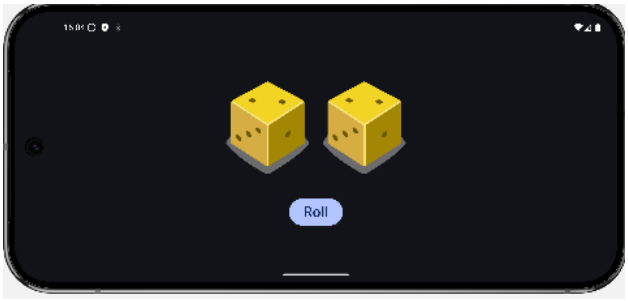
Gambar 4. Tampilan Aplikasi Jawaban saat Dibuka



Gambar 5. Tampilan Aplikasi saat Dadu yang Diacak Nilainya Berbeda



Gambar 6. Tampilan Aplikasi Jawaban Saat Dadu yang Diacak Nilainya Sama



Gambar 7. Tampilan Aplikasi Saat Dirotasikan

C. Pembahasan

1. MainActivity.kt:

Pertama, mendeklarasikan nama package-nya. Kemudian mengimpor librari yang dibutuhkan. Class `MainActivity` yang merupakan titik masuk utama aplikasi Android berbasis Jetpack Compose. Class ini meng-extend `ComponentActivity`, yang merupakan kelas dasar untuk aktivitas dalam Android yang mendukung Compose UI.

Pada metode `onCreate()`, yang dipanggil saat aktivitas pertama kali dibuat, dilakukan inisialisasi antarmuka pengguna dengan `setContent { ... }`. Di dalam blok ini, tema aplikasi (`DiceRollerComposeTheme`) diterapkan untuk memastikan konsistensi desain material. Selanjutnya, komponen `Surface` digunakan sebagai container utama yang mengisi seluruh layar (`Modifier.fillMaxSize()`) dengan warna latar belakang sesuai tema default dari librari `Material` (`MaterialTheme.colorScheme.background`).

Di dalam `Surface`, fungsi komposable `DiceRollerApp()` dipanggil untuk me-render seluruh UI aplikasi, termasuk tampilan dadu dan tombol roll. Aktivitas utama dari kode ini adalah mengatur tema dasar dan layout, sementara logika tampilan dan interaksi diimplementasikan dalam komponen `Composable` terpisah.

`DiceRollerApp()`, yang berperan sebagai komponen utama dalam aplikasi pemutar dadu. Anotasi `@Preview` memungkinkan fungsi ini ditampilkan di Android Studio's Preview Panel, agar dapat melihat pratinjau UI tanpa harus menjalankan aplikasi di emulator atau perangkat fisik. Anotasi `@Composable` digunakan agar suatu fungsi menjadi fungsi komposable (`composable function`). Fungsi ini digunakan untuk mendefinisikan bagian antarmuka pengguna (UI) secara deklaratif di Android.

Fungsi `DiceWithButtonAndImage` adalah komponen utama untuk menampilkan antarmuka pengguna yang interaktif dan dinamis. Fungsi ini menggunakan `DiceViewModel` untuk mengelola logika bisnis, seperti menyimpan nilai dadu kiri (`numDiceLeft`) dan kanan (`numDiceRight`), serta menghasilkan angka acak saat tombol ditekan. Dengan memanfaatkan `viewModel()`, instance `ViewModel` diambil secara otomatis, memastikan konsistensi state, bahkan selama perubahan konfigurasi seperti rotasi layar. Gambar dadu ditentukan melalui blok `when` yang memetakan nilai numerik dari `ViewModel` ke resource gambar yang sesuai (misalnya, `dice_1` untuk angka `1`). Jika terjadi nilai di luar rentang 1-6, gambar `dice_0` ditampilkan sebagai fallback, yang berfungsi sebagai indikator eror atau keadaan awal.

Antarmuka pengguna disusun secara hierarkis menggunakan `Column` sebagai container utama, yang mengatur elemen-elemen secara vertikal dan rata tengah horizontal (`Alignment.CenterHorizontally`). Di dalamnya, `Row` digunakan untuk menempatkan dua gambar dadu secara horizontal, dengan `Spacer` yang memberikan jarak `100dp` di antaranya. Setiap `Image` memiliki tinggi tetap `200dp` (`Modifier.height(200.dp)`) dan `contentDescription` yang dinamis berdasarkan nilai dadu. Di bawah `Row`, terdapat `Spacer` tambahan (`16dp`) dan `Button` yang menampilkan teks "Roll" dengan ukuran `24sp`.

Ketika tombol diklik, fungsi `onClick()` menjalankan dua operasi utama: memanggil `viewModel.rollDice()` untuk mengacak nilai kedua dadu, dan menampilkan `Toast` yang menampilkan pesan hasil lemparan dari `viewModel.getResultMessage()`. Saat nilai dadu di `ViewModel` berubah, fungsi komposable akan otomatis di-recompose untuk memperbarui gambar sesuai state terbaru, tanpa perlu pembaruan manual. `Toast` sendiri diposisikan di bagian bawah layar dengan `Gravity.BOTTOM or Gravity.CENTER_HORIZONTAL` dan offset vertikal `150px` untuk menghindari tumpang-tindih dengan navigasi sistem.

2. DiceViewModel.kt

Class `DiceViewModel` merupakan komponen inti dalam aplikasi yang mengadopsi pola Model-View-ViewModel (MVVM) untuk memisahkan logika bisnis dari antarmuka pengguna (UI). Class ini bertanggung jawab atas pengelolaan state (keadaan) aplikasi, logika penghitungan, dan interaksi data antara model dan tampilan.

Variabel `numDiceLeft` dan `numDiceRight` dideklarasikan menggunakan `mutableStateOf(0)`, yang menjadikan keduanya sebagai state yang dapat diamati (`observable state`). Dengan ini, Jetpack Compose dapat secara otomatis mendeteksi perubahan nilai dan memicu recomposition (pembaruan UI) hanya pada komponen yang bergantung pada state tersebut. Nilai awal 0 dipilih sebagai penanda bahwa dadu belum dikocok, yang ditampilkan sebagai gambar dadu kosong (`dice_0`). Penggunaan `private set` pada kedua variabel membatasi modifikasi nilai hanya melalui fungsi di dalam `ViewModel` (seperti `rollDice()`), sehingga memastikan enkapsulasi data dan mencegah perubahan tidak sah dari luar kelas.

Fungsi `getResultMessage()` mengembalikan pesan hasil berdasarkan kesamaan nilai kedua dadu. Jika `numDiceLeft` sama dengan `numDiceRight`, pesan "Selamat, Anda mendapatkan nilai double" dikirim, yang mengindikasikan hasil khusus (seperti double dalam permainan dadu). Jika tidak, pesan "Maaf, Anda belum beruntung" ditampilkan. Pesan ini kemudian digunakan di layer UI (`Toast`).

`ViewModel` ini diakses di komponen UI (`DiceWithButtonAndImage`) melalui fungsi `viewModel()`, yang menjamin instance `ViewModel` tetap bertahan selama siklus hidup aktivitas (activity lifecycle) atau fragment. Hal ini mencegah kehilangan data saat terjadi perubahan konfigurasi (misalnya, rotasi layar).

Saat tombol “Roll” diklik di UI:

a) `rollDice()` dipanggil untuk memperbarui state.
b) Perubahan nilai `numDiceLeft` dan `numDiceRight` memicu recomposition pada `Image` yang terkait.
c) `getResultMessage()` mengambil pesan hasil untuk ditampilkan di `Toast`.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat
https://github.com/MinamotoYuki46/MeineStudienArbeit/tree/3f3003c145eb3e6f33f623fc5d8885dde05a385e/MobileDevelopment/Codex-Practicus/Modul%201/Jetpack%20Compose/DiceRollerCompose