LAPORAN PRAKTIKUM
PEMROGRAMAN MOBILE
MODUL 1

ANDROID BASIC WITH KOTLIN

Oleh:

Alysa Armelia **NIM. 2310817120009**

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 : Alysa Armelia
NIM : 2310817120009

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

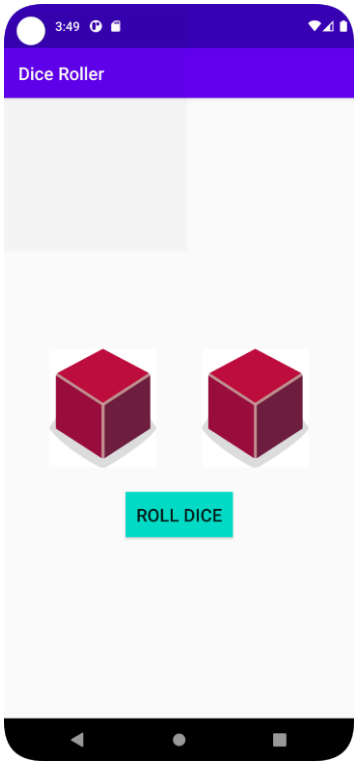
DAFTAR TABEL

SOAL 1

Soal Praktikum:

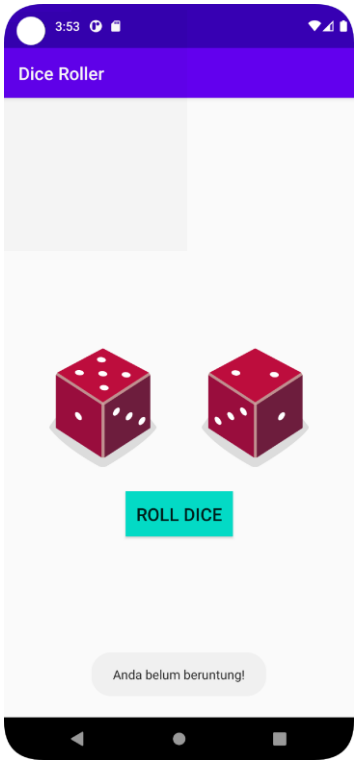
Buatlah sebuah aplikasi yang dapat menampilkan 2 (dua) buah dadu yang dapat berubah-ubah tampilannya pada saat user menekan tombol “Roll Dice”. Aturan aplikasi yang akan dibangun adalah sebagaimana berikut:

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.

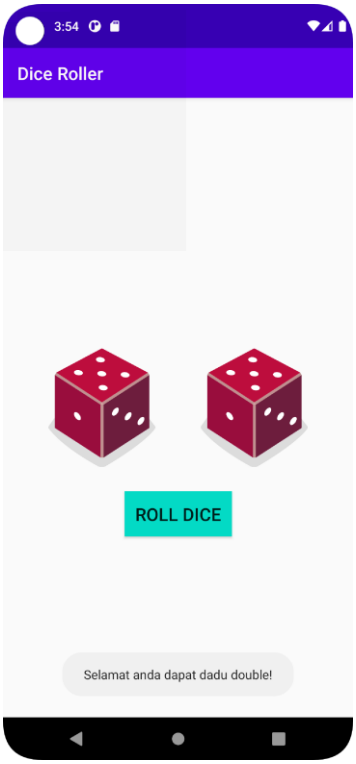


Gambar 2 Tampilan Dadu Setelah Di Roll

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

1. MainActivity.kt

```
1   package com.example.diceroller
2
3   import android.os.Bundle
4   import android.widget.Toast
5   import androidx.activity.viewModels
6   import androidx.appcompat.app.AppCompatActivity
7   import androidx.lifecycle.Observer
8   import com.example.diceroller.databinding.ActivityMainBinding
9
10  class MainActivity : AppCompatActivity() {
11
12      private lateinit var binding: ActivityMainBinding
13      private val viewModel: Dice_ViewModel by viewModels()
14
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17
18          binding = ActivityMainBinding.inflate(layoutInflater)
19          setContentView(binding.root)
20
21          binding.button.setOnClickListener {
```

```
22              viewModel.rollDice()
23          }
24
25          viewModel.dice1.observe(this) { value ->

26  binding.imageView.setImageResource(getDiceImage(value))
27          }
28
29          viewModel.dice2.observe(this) { value ->

30  binding.imageView2.setImageResource(getDiceImage(value))
31          }
32
33          viewModel.isDouble.observe(this) { isDouble ->
34              val message = if (isDouble) {
35                  "Selamat anda dapat dadu double!"
36              } else {
37                  "Anda belum beruntung!"
38              }
39              Toast.makeText(this, message,
    Toast.LENGTH_SHORT).show()
40          }
41      }
42
43      private fun getDiceImage(number: Int): Int {
44          return when (number) {
45              1 -> R.drawable.dice_1
46              2 -> R.drawable.dice_2
47              3 -> R.drawable.dice_3
48              4 -> R.drawable.dice_4
49              5 -> R.drawable.dice_5
50              else -> R.drawable.dice_6
51          }
52      }
53  }
```

Tabel 1. 1 Source Code Jawaban Soal 1

2. Dice_ViewModel.kt

```
1   package com.example.diceroller
2
3   import androidx.lifecycle.LiveData
4   import androidx.lifecycle.MutableLiveData
5   import androidx.lifecycle.ViewModel
6
7   class Dice_ViewModel : ViewModel() {
8
9       private val _dice1 = MutableLiveData(1)
10      val dice1: LiveData<Int> = _dice1
11
```

```
12      private val _dice2 = MutableLiveData(1)
13      val dice2: LiveData<Int> = _dice2
14
15      private val _isDouble = MutableLiveData(false)
16      val isDouble: LiveData<Boolean> = _isDouble
17
18      fun rollDice() {
19          val d1 = (1..6).random()
20          val d2 = (1..6).random()
21          _dice1.value = d1
22          _dice2.value = d2
23          _isDouble.value = (d1 == d2)
24      }
25  }
```

Tabel 1. 2 Source Code Jawaban Soal 1

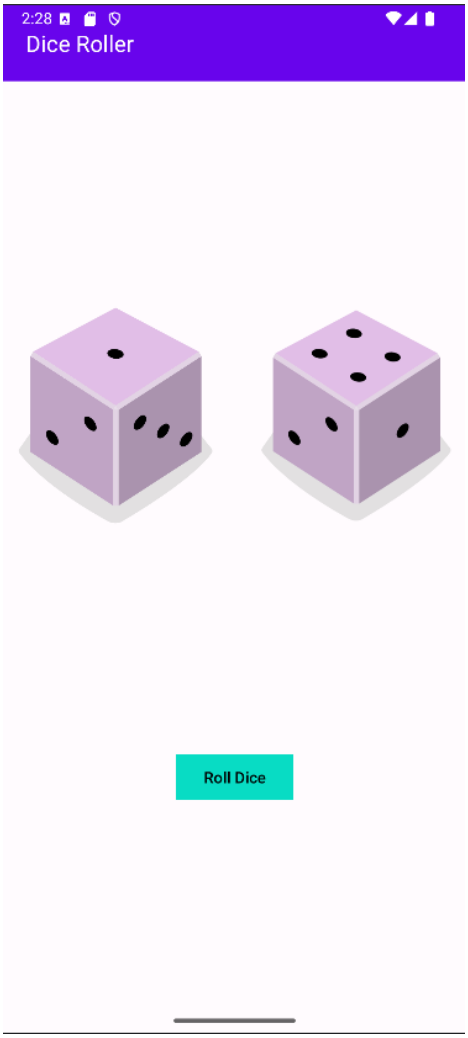
3. activity_main.xml

```
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:id="@+id/main"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       tools:context=".MainActivity">
9
10      <TextView
11          android:id="@+id/headerTitle"
12          android:layout_width="0dp"
13          android:layout_height="wrap_content"
14          android:background="#6804ec"
15          android:padding="20dp"
16          android:text="@string/app_name"
17          android:textColor="#FFFFFF"
18          android:textSize="20sp"
19          app:layout_constraintEnd_toEndOf="parent"
20          app:layout_constraintHorizontal_bias="0.0"
21          app:layout_constraintStart_toStartOf="parent"
22          app:layout_constraintTop_toTopOf="parent" />
23
24      <LinearLayout
25          android:id="@+id/diceContainer"
26          android:layout_width="wrap_content"
27          android:layout_height="wrap_content"
28          android:orientation="horizontal"
29          android:gravity="center"
30  app:layout_constraintTop_toBottomOf="@+id/headerTitle"
31          app:layout_constraintBottom_toTopOf="@+id/button"
```

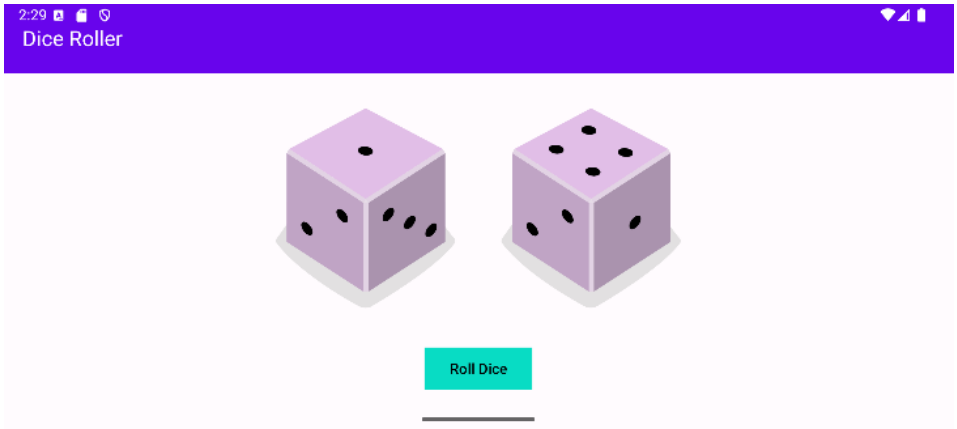
```
32          app:layout_constraintStart_toStartOf="parent"
33          app:layout_constraintEnd_toEndOf="parent"
34          app:layout_constraintVertical_bias="0.5">
35
36          <ImageView
37              android:id="@+id/imageView"
38              android:layout_width="wrap_content"
39              android:layout_height="wrap_content"
40              android:src="@drawable/dice_0" />
41
42          <ImageView
43              android:id="@+id/imageView2"
44              android:layout_width="wrap_content"
45              android:layout_height="wrap_content"
46              android:src="@drawable/dice_0"
47              android:layout_marginStart="16dp" />
48
49      </LinearLayout>
50
51      <com.google.android.material.button.MaterialButton
52          android:id="@+id/button"
53          android:layout_width="wrap_content"
54          android:layout_height="wrap_content"
55          android:text="@string/button_1"
56          android:backgroundTint="#08dcc4"
57          android:textColor="@color/black"
58          app:cornerRadius="0dp"
59    app:layout_constraintTop_toBottomOf="@+id/diceContainer"
60          app:layout_constraintBottom_toBottomOf="parent"
61          app:layout_constraintStart_toStartOf="parent"
62          app:layout_constraintEnd_toEndOf="parent"
63          app:layout_constraintVertical_bias="0.0"
64          android:layout_marginBottom="32dp" />
65
66  </androidx.constraintlayout.widget.ConstraintLayout>
```

Tabel 1. 3 Source Code Jawaban Soal 1

B. Output Program



Gambar 1 Screenshot Hasil Jawaban Soal 1



Gambar 2 Screenshot Hasil Jawaban Soal 1

C. Pembahasan

1. MainActivity.kt

Pada line 1-8 kita perlu melakukan deklarasikan nama package file Kotlin agar dapat menentukan lokasi file sesuai struktur project. Dimana disini ada `import` yang berguna saat kita ingin mengimpor kelas-kelas dibutuhkan seperti, `Toast` untuk menampilkan pesan pop-up, `AppCompatActivity` sebagai ctivity dasar dalam Android, `viewModels()` untuk mengambil ViewModel, dan `ActivityMainBinding` untuk mengakses elemen layout menggunakan View Binding. Seperti di line [10], [12], [13], dan [53] kita perlu mendeklarasikan class dan preoperti. Yang mana disini ada class `MainActivity` merupakan activity utama atau untuk bagian halaman utama aplikasi, `binding` berguna saat menghubungkan layout XML dengan kode Kotlin, juga `viewModel` membantu menghubungkan ke class `Dice_ViewModel` yang berisi logika bagaimana cara melempar dadu tersebut. Lalu, di line [15], [16], dan [41] ada `onCreate()` karena fungsi ini dipanggil saat activity dibuat pertama kali dengan `savedInstanceState` berguna untuk menyimpan data saat kita melakukan rotasi layar, dll. Kemudian, di line [18] – [19] perlu melakukan inisialisasi layout karena menggunakan View Binding untuk mempermudah akses elemen UI jangan lupa disini ada `setContentView(binding.root)` berguna saat menampilkan layout XML ke layar. Selanjutnya, di line [21] – [22] adalah tombol untuk melempar dadu saat tombol tersebut ditekan nantinya fungsi `rollDice()` bagian `ViewModel` dipanggil. Dengan fungsi ini juga akan mengacak dua angka dadu dan update `LiveData`. Dilihat dari line [25] – [31] adalah kode agar bisa mengamati LiveData hal ini berguna untuk mengamati perubahan nilai `dice1` karena saat nilainya berubah, maka gambar dadu pertama (`imageView`) akan diperbarui sesuai angka didapat juga berlaku sama seperti kode di atas bagian `dice2` karena untuk dadu kedua (`imageView2`). Ada line [33] – [40] kita perlu menampilkan pesan `Toast` untuk `Double` karena penggunaan `isDouble` adalah `LiveData<Boolean>` dari `ViewModel`, jika nilainya `true` (kedua dadu sama) yang ditampilkan pesan keberuntungan dan juga berlaku saat tidak sama akan ditampilkan pesan anda belum

beruntung, serta `Toast.makeText(...)` berguna menampilkan pesan pendek di bawah layar. Setelah itu, line [43] – [52] berguna sebagai fungsi untuk mengubah angka jadi gambar karena fungsi ini akan mengembalikan ID gambar `R.drawable.dice_Xs` sesuai angka dadu (1-6) dengan bagian `observe` berperan untuk mengganti gambar dadu.

2. Dice_ViewModel.kt

Pada line [1] – [5] kita perlu melakukan deklarasikan nama package file Kotlin agar dapat menentukan lokasi file sesuai struktur project. Dimana disini ada `import` yang berguna saat kita ingin mengimpor kelas-kelas dibutuhkan seperti, `LiveData`, `MutableLiveData` berguna untuk mengamati data secara otomatis dari UI, juga ada `ViewModel` adalah class dari Android Architecture Component untuk menyimpan dan mengelola data UI. Kemudian, di line [7] dan [25] kita perlu melakukan deklarasi class `ViewModel` karena `Dice_ViewModel` bagian dari `ViewModel` khusus untuk aplikasi ini bertujuan dalam menyimpan data serta logika melempar dadu agar tetap bertahan walaupun kita merotasi layarnya. Lalu, di line [9] – [13] LiveData untuk dadu pertama yang mana `_dice1` agar menyimpan angka dadu pertama secara default nilainya 1 melibatkan `MutableLiveData` untuk dapat dirubah dari dalam ViewModel dengan `dice1` akan dibuka sebagai `LiveData` memudahkan saat mengamati dari segi UI (tapi tidak bisa diubah dari luar). Juga disini mungkin pasti bingung kenapa dibagi dua (`_dice1` dan `dice1`)? karena ada enkapsulasi supaya hanya ViewModel yang bisa mengubah nilai dan bagian UI hanya bisa membaca, hal ini berlaku LiveData untuk dadu kedua agar menyimpan angka dadu kedua. Selanjutnya, di line [15] – [16] adalah LiveData untuk deteksi double maksudnya akan menyimpan nilai `true` apabila kedua dadu nilainya sama dan nilai ini digunakan di UI agar bia menampilkan `toast` keberuntungan. Berikutnya, di line [18] – [24] ada fungsi `rollDice()` yang akan mengacak dua angka antara 1 sampai 6 dengan menyimpan hasilnya ke `_dice1` dan `_dice2` dan perlu di cek lagi apakah hasilnya sama, jika ya maka set `_isDouble` ke `true` nantinya `LiveData` secara otomatis memberi tahu UI saat nilainya berubah.

3. activity_main.xml

Pada line [2] – [8] terdapat struktur utama layout terdiri dari `ConstraintLayout` merupakan Layout fleksibel yang bisa memberikan kemungkinan ke kita bagimana cara mengatur posisi elemen UI berdasarkan hubungan antar elemen melibatkan `match_parent` untuk lebar dan tinggi layout mengikuti ukuran layar penuh dengan `tools:context=".MainActivity"` agar menunjukkan layout ini digunakan oleh `MainActivity`. Kemudian, lihat line [10] – [22] berguna sebagai `TextView` di Judul Aplikasi yang mana agar bisa menampilkan judul aplikasi menggunakan `@string/app_name` dari `strings.xml` dengan ukuran lebar `0dp` berarti mengikuti batas constraint kiri dan kanan (dari parent), `padding="20dp"` yaitu jarak dalam elemen, `background` ungu dan teks putih, juga posisi itu tempel ke bagian atas layar atau `Top_toTopOf="parent"`, serta menempel ke kiri dan kanan layar (Start & End). Lalu, di line [24] – [49] ada `LinearLayout` sebagai wadah gambar dadu disini nantinya berisi dua buah gambar dadu secara horizontal dengan `wrap_content` untuk ukurannya mengikuti isi melibatkan `gravity="center"` menjadi isi elemen diposisikan di Tengah dan posisi terletak di bawah `headerTitle`, di atas tombol atau `button`, dan diletakkan di tengah vertikal (`vertical_bias="0.5"` = tengah antara atas dan bawah). Selanjutnya, line [36] – [40] ada `ImageView` untuk menampilkan gambar dadu pertama yang mana gambar awal adalah `dice_0`. Berikutnya, line [42] – [47] ada `ImageView 2` untuk menampilkan gambar dadu kedua dengan diberi jarak ke kanan ukurannya `16dp` dari gambar dadu pertama agar tidak menempel. Setelah itu, line [51] – [64] ada `MaterialButton` adalah tombol digunakan untuk lempar dadu, teks diambil dari `strings.xml` atau `@string/button_1` yang warna latar tombol itu cyan (`#08dcc4`) menggunkan `cornerRadius="0dp"` agar tombol berbentuk kotak dengan posisi di bawah `diceContainer` dan tempel ke bagian bawah layar (`Bottom_toBottomOf="parent"`) ukuran margin `32dp`.

D. Tautan Git

Berikut adalah tautan untuk source code yang telah dibuat.

https://github.com/alysaarmelia/AlysaArmelia_2310817120009_Pemrograman_Mobile/tree/2b8d8eda77d172d131905c595b0e6a59af202137/PRAK_MODUL1