

Lab 2

Disusun Oleh:

Usamah Nashirul Haq - 1606917954

28 September 2020

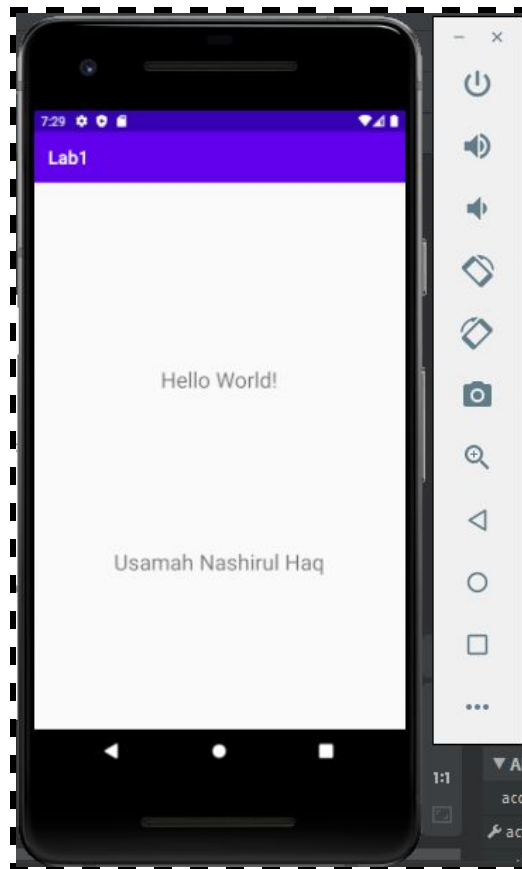
<https://github.com/usamah1707/learn-ktkpl-1606917954>

A. Deskripsi Aplikasi

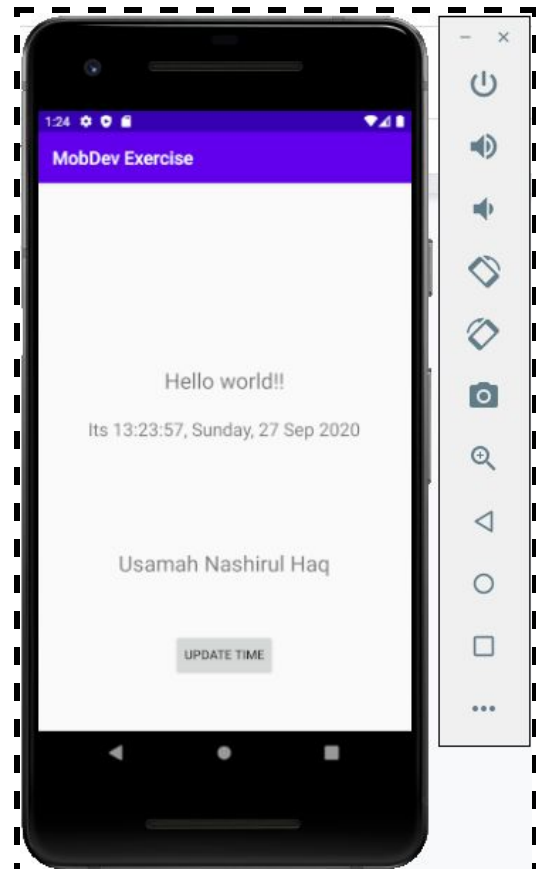
Aplikasi yang dibuat pada Lab 2 kali ini adalah aplikasi *hello world* yang memanfaatkan *unit testing* dan *instrumented testing*. Aplikasi ini dibuat untuk mencoba penggunaan fungsi *testing* pada pengembangan aplikasi android. Aplikasi *hello world* ini juga telah dikembangkan dengan menambahkan beberapa fungsi lain seperti tombol untuk meng-*update* tanggal. Inti dari aplikasi kali ini hanyalah melakukan *unit testing (local testing)* dan *instrumented testing* di *android virtual device* atau pada perangkat yang dihubungkan dengan Android Studio oleh tiap-tiap mahasiswa pada saat aplikasi ini dijalankan.

B. Langkah Pengerjaan

Pada bagian ini, saya akan menjelaskan tahapan yang saya lakukan untuk mengembangkan aplikasi *Hello World* yang dibuat pada Lab 1 pada Android Studio dengan menambahkan *unit testing* dan *instrumented testing*. Sebelumnya, saya sudah membuat aplikasi sederhana ini dengan menampilkan tulisan "Hello World" saja.



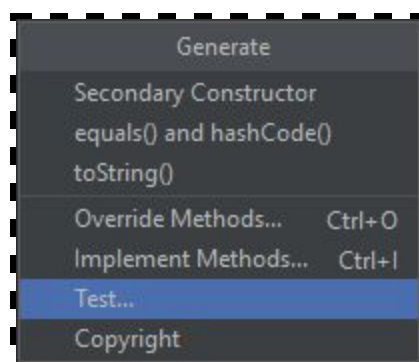
Versi Lab 1



Versi Lab 2

1. Unit Testing

Pada dasarnya, pembuatan *unit testing* pada Android Studio cukup mudah. Dimulai dengan membuka class yang akan di-test. Pada bagian teks editor, klik **alt + insert**, pilih **Test** untuk membuat *file unit testing* baru.



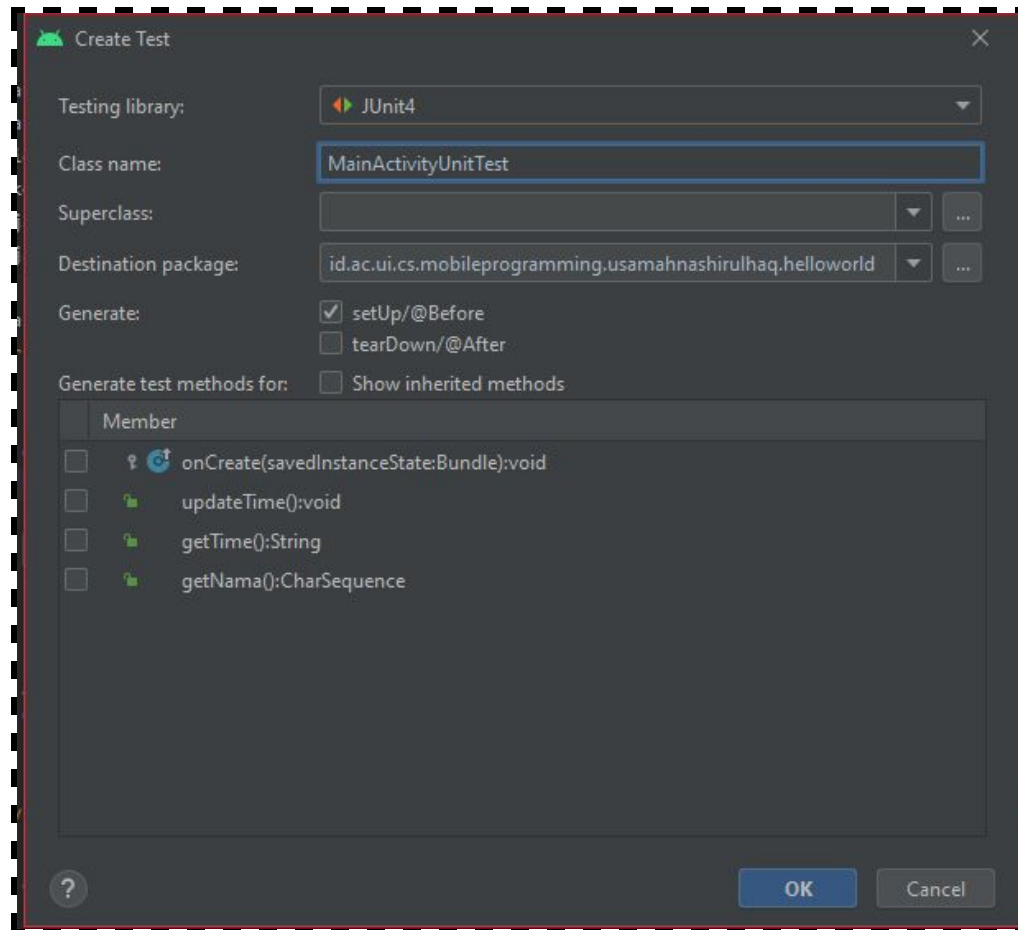
Setelah kita memilih Test pada menu Generate, akan muncul *window* seperti berikut. Disini kita perlu melakukan pengaturan untuk *file unit test* yang akan kita buat. Kita perlu mengisi *testing library*, *class name*, *superclass* (opsional), *destination*

Lab 2

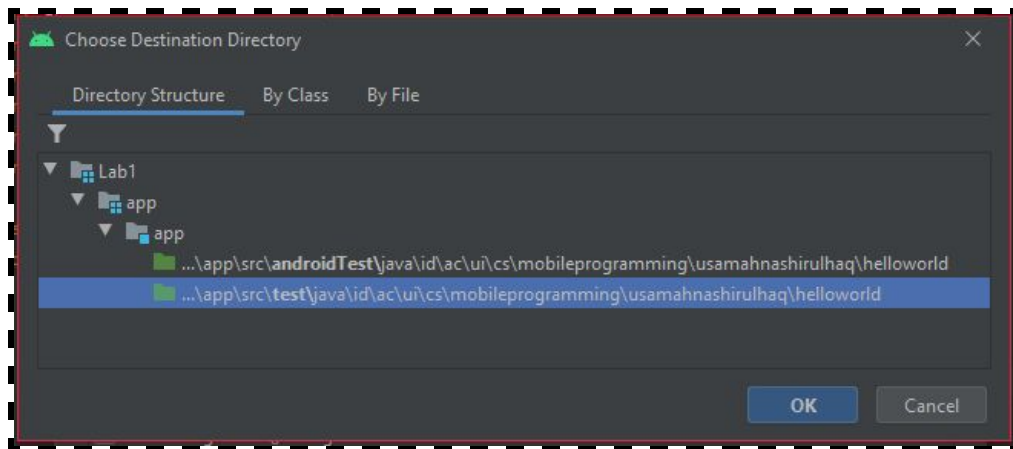
Usamah Nashirul Haq - 1606917954

<https://github.com/usamah1707/learn-tktp1-1606917954>

package, dan pilihan *auto generate* (opsional) untuk fungsi-fungsi tertentu. Saya mengisi *testing library* dengan **JUnit4**. Terdapat JUnit5 yang merupakan versi terbaru, namun tidak saya pilih mengingat kestabilan JUnit4 yang sudah lebih dulu teruji. Saya memberi nama MainActivityUnitTest sebagai *class name*. Saya juga memilih untuk membuat fungsi **setUp/@Before** yang dibuat secara otomatis untuk *file testing* saya. Sisanya saya biarkan *default* apa adanya. Selanjutnya klik **OK**.



Akan muncul satu *window* lagi yang meminta kita untuk memberikan lokasi *directory* yang kita inginkan. Karena yang kita buat adalah *unit test*, maka kita akan menempatkannya di **folder test**, bukan **androidTest**. Ini penting karena perbedaan lokasi yang kita pilih untuk menyimpan *file* ini akan menentukan jenis tes tersebut. *Folder androidTest* digunakan untuk **menyimpan instrumented test**. Selanjutnya klik **OK**.



Berikut adalah wujud pertama dari *file* MainActivityUnitTest. Selanjutnya saya akan mengisi *file* ini dengan beberapa tes.

```

1 package id.ac.ui.cs.mobileprogramming.usamahnashirulhaq.helloworld
2
3
4 import org.junit.Before
5
6 import org.junit.Assert.*
7 import java.text.SimpleDateFormat
8 import java.util.*
9
10 class MainActivityUnitTest {
11
12     @Before
13     fun setUp() {
14     }
15 }

```

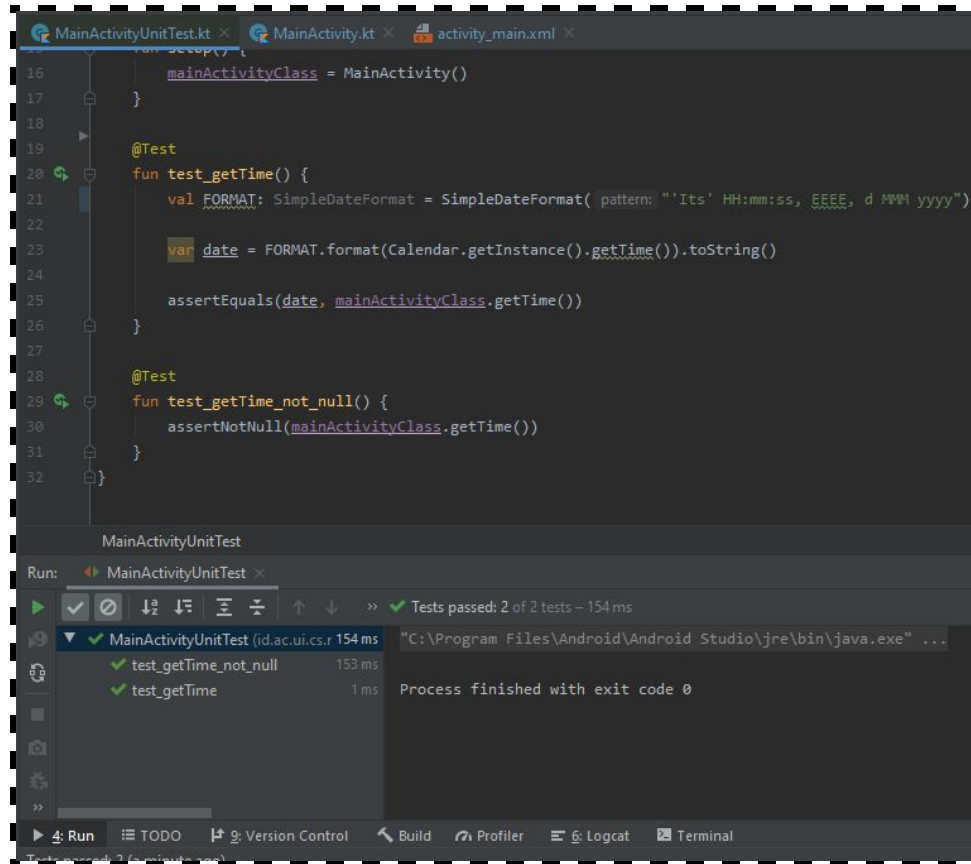
Untuk melakukan *unit test*, saya sudah mengubah MainActivity *class* saya agar memiliki sesuatu untuk dites. Berikut adalah wujud MainActivity *class* saya.

```
MainActivityUnitTest.kt x MainActivity.kt x activity_main.xml x
1 package id.ac.ui.cs.mobileprogramming.usamahnashirulhaq.helloworld
2
3 import ...
9
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14         dateView.setText(getTime())
15         buttonCheckTime.setOnClickListener() { it: View!
16             updateTime()
17         }
18     }
19
20     fun updateTime() {
21         dateView.setText(getTime())
22     }
23
24     private val FORMAT: SimpleDateFormat = SimpleDateFormat( pattern: "'It's' HH:mm:ss, EEEE, d MMM yyyy")
25
26     fun getTime(): String? {
27         return FORMAT.format(Calendar.getInstance().getTime()).toString()
28     }
29
30     fun getNama(): CharSequence? {
31         return getText(R.string.nama)
32     }
33 }
```

Dan ini adalah wujud MainActivityUnitTest setelah saya menerapkan tes-tes yang baru.

```
MainActivityUnitTest.kt x MainActivity.kt x activity_main.xml x
1 package id.ac.ui.cs.mobileprogramming.usamahnashirulhaq.helloworld
2
3 import ...
9
10 class MainActivity : AppCompatActivity() {
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14         dateView.setText(getTime())
15         buttonCheckTime.setOnClickListener() { it: View!
16             updateTime()
17         }
18     }
19
20     fun updateTime() {
21         dateView.setText(getTime())
22     }
23
24     private val FORMAT: SimpleDateFormat = SimpleDateFormat( pattern: "'It's' HH:mm:ss, EEEE, d MMM yyyy")
25
26     fun getTime(): String? {
27         return FORMAT.format(Calendar.getInstance().getTime()).toString()
28     }
29
30     fun getNama(): CharSequence? {
31         return getText(R.string.nama)
32     }
33 }
```

Setelah saya jalankan tesnya, saya berhasil mendapatkan status **Test passed**.



```
16 mainActivityClass = MainActivity()
17 }
18
19 @Test
20 fun test_getTime() {
21     val FORMAT: SimpleDateFormat = SimpleDateFormat( pattern: "'Its' HH:mm:ss, EEEE, d MMM yyyy")
22
23     var date = FORMAT.format(Calendar.getInstance().getTime()).toString()
24
25     assertEquals(date, mainActivityClass.getTime())
26 }
27
28 @Test
29 fun test_getTime_not_null() {
30     assertNotNull(mainActivityClass.getTime())
31 }
32 }
```

MainActivityUnitTest

Run: MainActivityUnitTest

Tests passed: 2 of 2 tests - 154 ms

Test Name	Duration
MainActivityUnitTest (id.ac.uics.r)	154 ms
test_getTime_not_null	153 ms
test_getTime	1 ms

Process finished with exit code 0

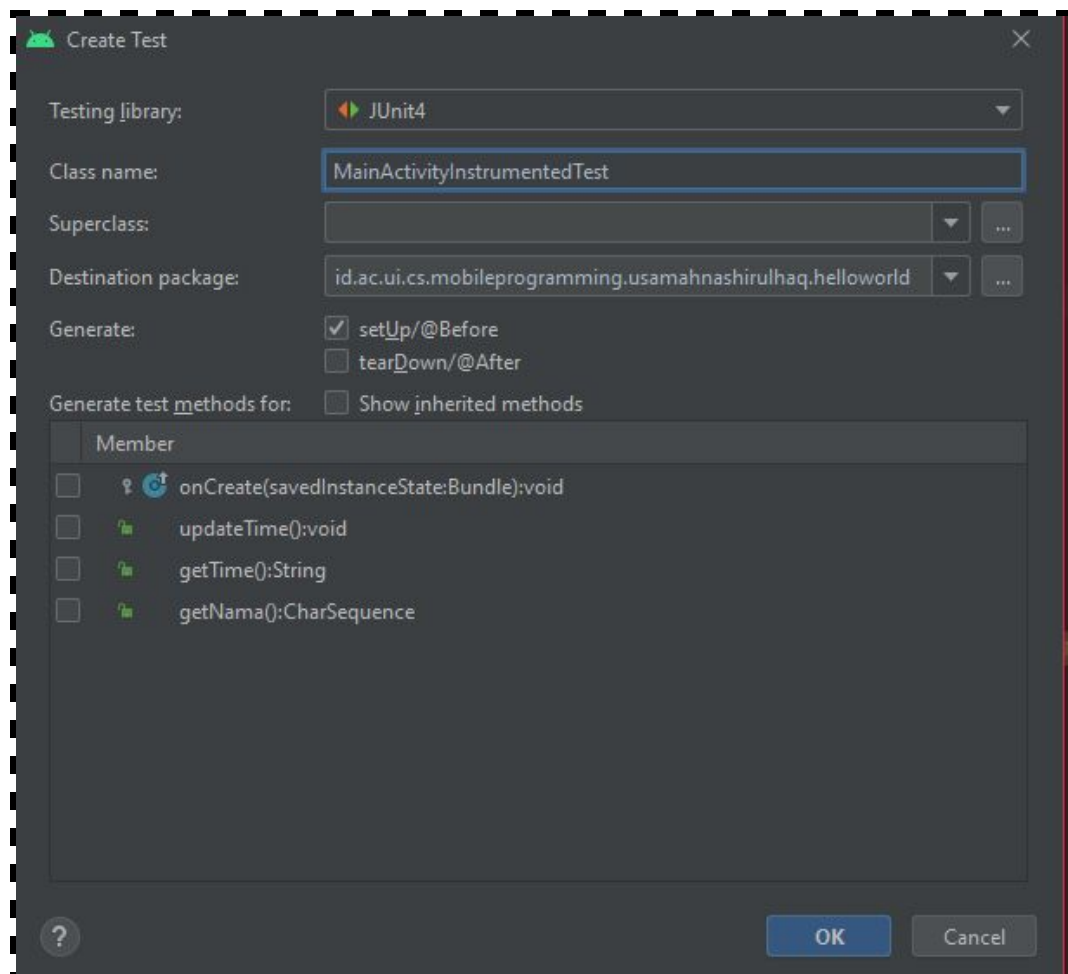
Kendala dan kesulitan: Untuk memulai *unit test* pertama kali tentunya saya bingung terkait apa yang perlu dites dalam *unit test*. Saya juga bingung bagaimana memanggil komponen-komponen yang akan saya test dari MainActivity.kt dan activity_main.xml ke dalam *file testing*. Meskipun saya sudah berhasil memasukan dan melakukan *testing* pada komponen-komponen tersebut, saya masih belum yakin dengan apa yang telah saya lakukan karena keterbatasan ilmu yang masih saya miliki.

Penyelesaian masalah: Masalah seperti ini saya rasa sangat umum dirasakan oleh para pemula seperti saya. Hal ini dikarenakan ilmu dan pengalaman yang masih sempit membuat kami kebingungan. Untuk menyelesaikan masalah seperti ini, saya harus menonton video tutorial dan penjelasan di Youtube. Saya juga membaca berbagai artikel android mulai dari dokumentasi *developer*, *stackoverflow*, hingga *medium*. Saya juga melakukan diskusi dengan teman-teman kolaborator saya untuk bertukar pikiran dan saling mengusulkan solusi untuk tiap-tiap masalah yang muncul. Meskipun pada akhirnya saya dapat sedikit membayangkan apa yang sedang dan akan saya lakukan, saya merasa hal tersebut masih belum cukup untuk membuat saya paham dan tahu bagaimana caranya untuk melakukan *unit test* dengan benar dan profesional.

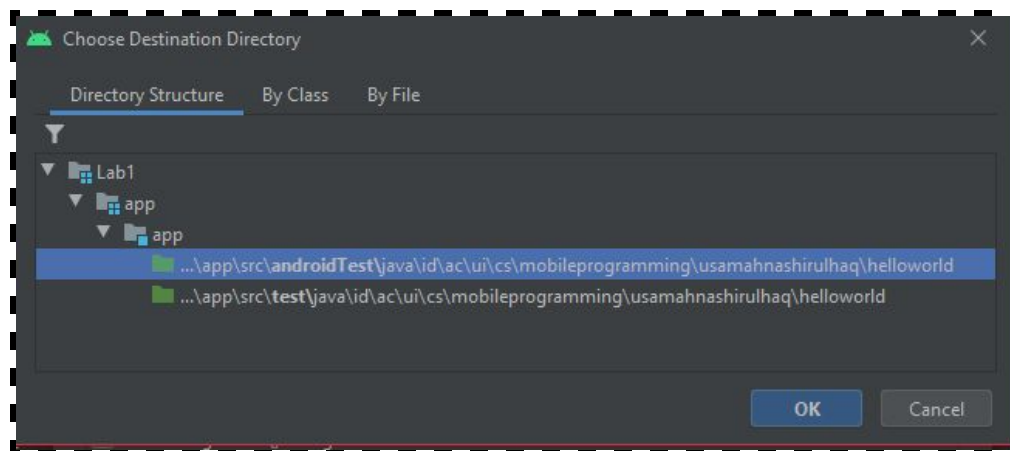
2. Instrumented Test

Pada dasarnya pembuatan *instrumented test* relatif sama dengan cara membuat *unit test*. Yang membedakan adalah dimana *directory* yang akan menyimpan *instrumented test* ini. Kita mulai langsung dari tahap pengaturan *file testing* yang akan kita buat.

Saya mengisi *testing library* dengan **JUnit4**. Saya memberi nama MainActivityInstrumentedTest sebagai *class name*. Saya juga memilih untuk membuat fungsi **setUp/@Before** yang dibuat secara otomatis untuk *file testing* saya. Sisanya saya biarkan *default* apa adanya. Selanjutnya klik **OK**.



Akan muncul satu *window* lagi yang meminta kita untuk memberikan lokasi *directory* yang kita inginkan. Karena yang kita buat adalah *instrumented test*, maka kita akan menempatkannya di **folder androidTest**, bukan **test**. Selanjutnya klik **OK**.



Dan ini adalah wujud MainActivityInstrumentedTest setelah saya menerapkan tesnya.

```
class MainActivityInstrumentedTest {
    @get:Rule
    var activityTestRule : ActivityTestRule<MainActivity!> = ActivityTestRule(MainActivity::class.java)
    private lateinit var mActivity: MainActivity

    @Before
    fun setUp() {
        mActivity = activityTestRule.activity
    }

    @Test
    fun test_mainActivity() {
        assertNotNull(mActivity)
    }

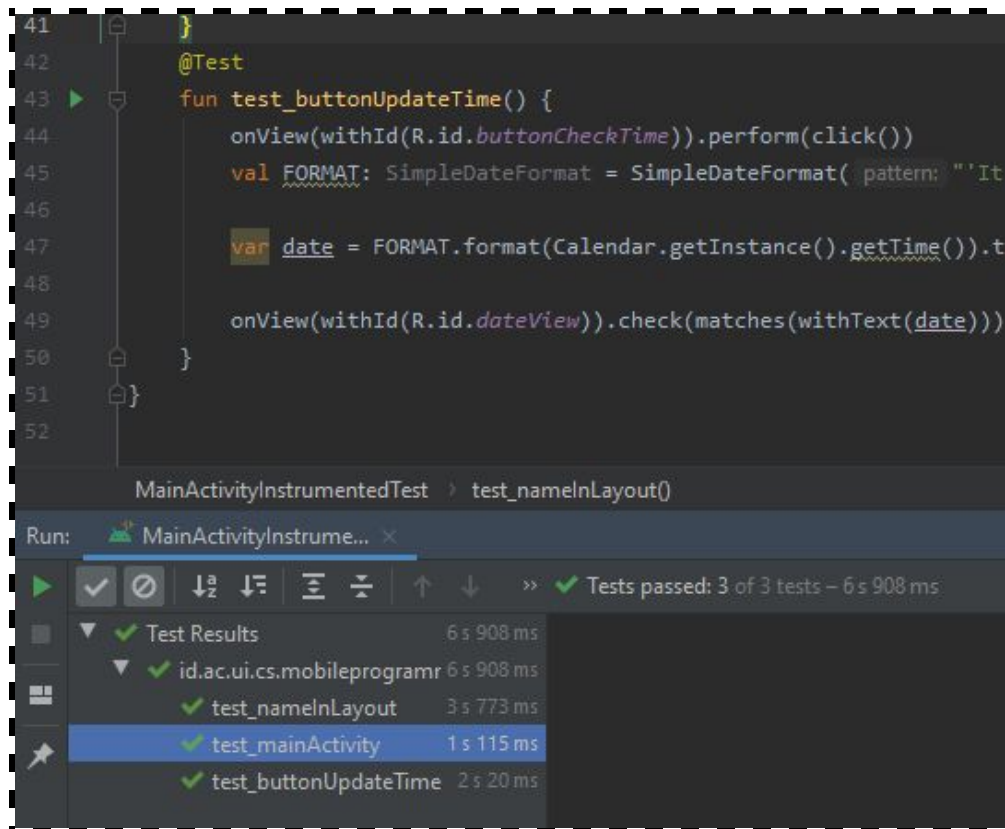
    @Test
    fun test_nameInLayout() {
        var name = mActivity.getNama()
        assertEquals( expected: "Usamah Nashirul Haq", name)
    }

    @Test
    fun test_buttonUpdateTime() {
        onView(withId(R.id.buttonCheckTime)).perform(click())
        val FORMAT: SimpleDateFormat = SimpleDateFormat( pattern: "'Its' HH:mm:ss, EEEE, d MMM yyyy")

        var date = FORMAT.format(Calendar.getInstance().getTime()).toString()

        onView(withId(R.id.dateView)).check(matches(withText(date)))
    }
}
```

Setelah saya jalankan tesnya, saya berhasil mendapatkan status **Test passed**.

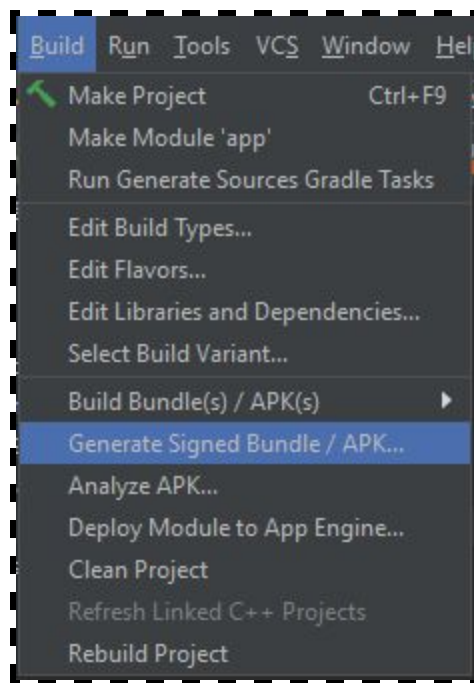


Kendala dan kesulitan: Kendala dan kesulitan yang saya alami pada bagian *instrumented test* kurang lebih sama dengan yang ada pada *unit test*. Namun pada bagian *instrumented test* saya menyadari bahwa terdapat perbedaan mendasar antara *unit test* dengan *instrumented test*. Masalah ini mulai membawa saya kepada kebingungan terkait mencari contoh *instrumented test* yang tepat seperti apa. Masalah lain seperti *dependencies* yang dibutuhkan dan *library* apa yang perlu saya gunakan.

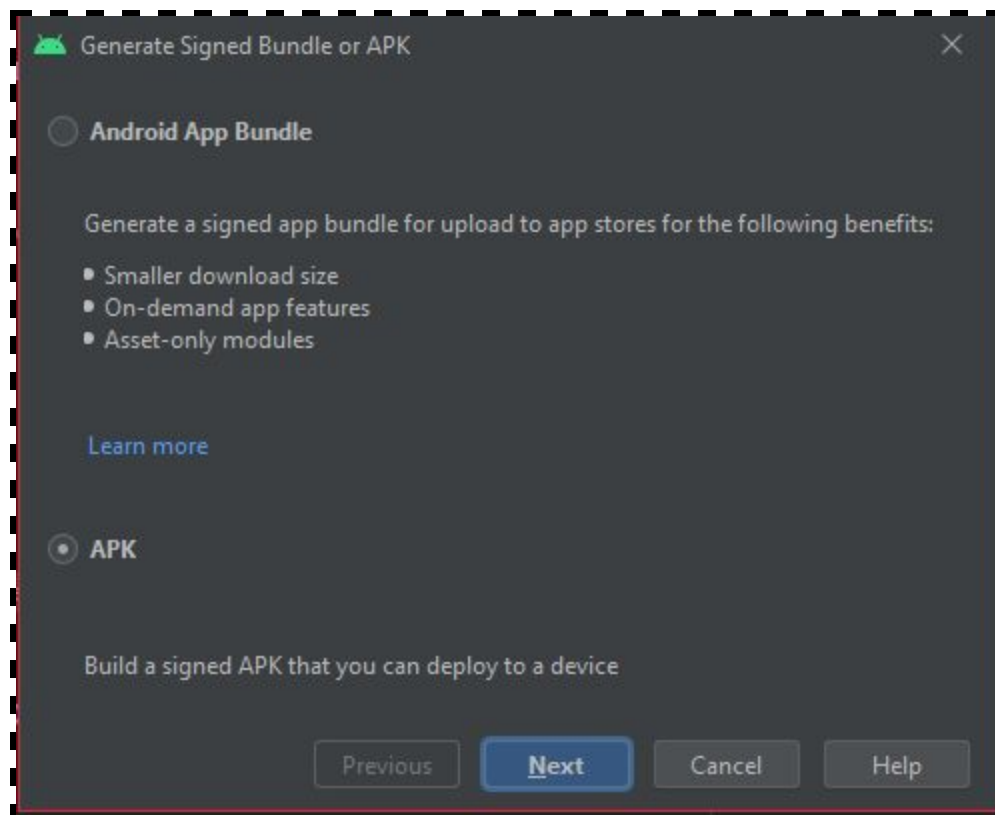
Penyelesaian masalah: Untuk menyelesaikan masalah seperti ini, saya harus menonton video tutorial dan penjelasan di Youtube. Saya juga membaca berbagai artikel android mulai dari dokumentasi *developer*, *stackoverflow*, hingga *medium*. Saya juga melakukan diskusi dengan teman-teman kolaborator saya untuk bertukar pikiran dan saling mengusulkan solusi untuk tiap-tiap masalah yang muncul. Sama seperti penyelesaian masalah di bagian *unit test*, saya merasa masih belum cukup paham dan tahu bagaimana caranya untuk melakukan *unit test* dengan benar dan profesional.

3. Build Signed APK

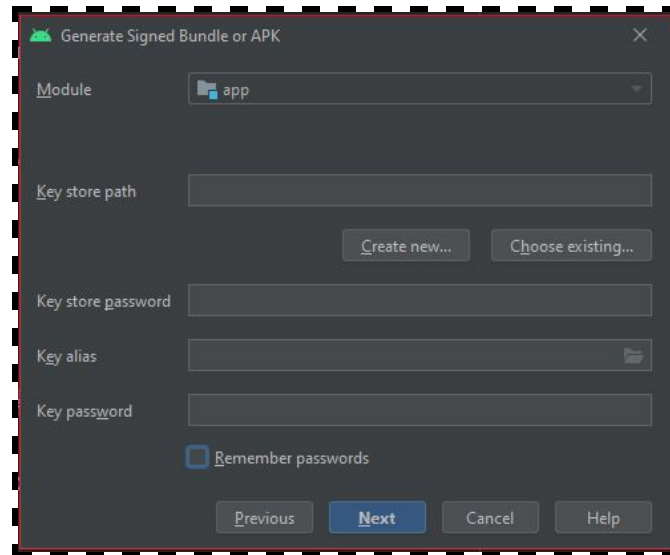
Untuk melakukan *build signed* APK, buka **Build > Generate Signed Bundle/APK**.



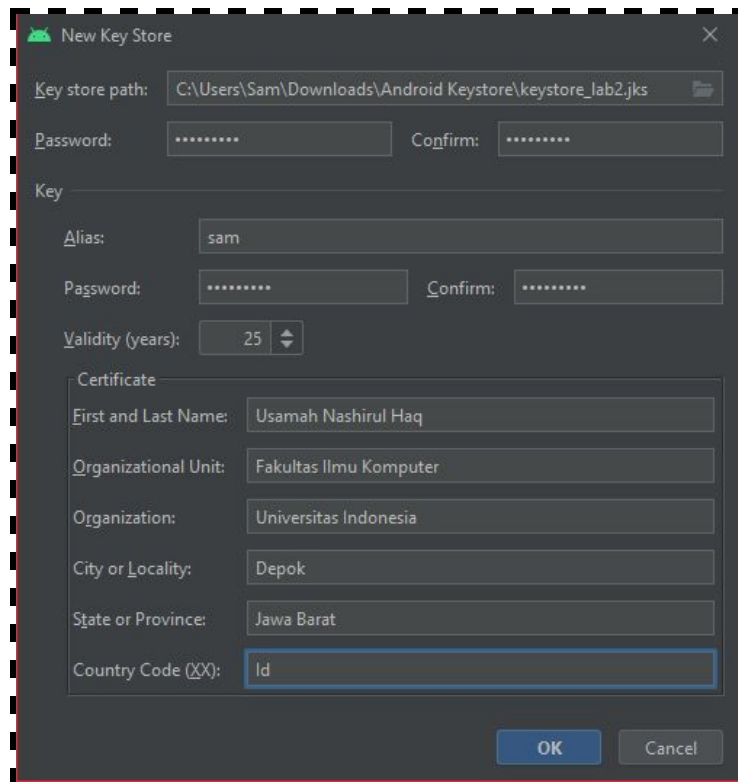
Setelah itu akan muncul *window* dengan tampilan sebagai berikut. Pilih **APK**, lalu klik **Next**.



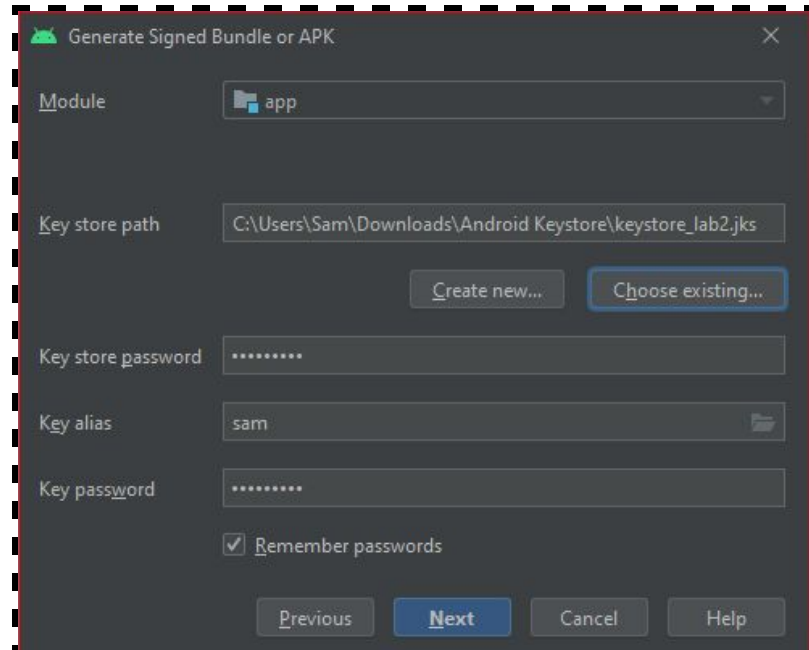
Akan muncul tampilan seperti berikut. Terdapat beberapa bagian yang perlu diisi, namun dengan menekan tombol **Create new** pada bagian **Key store path**, kita bisa mengisi semua kolom tersebut. Klik **Create new**.



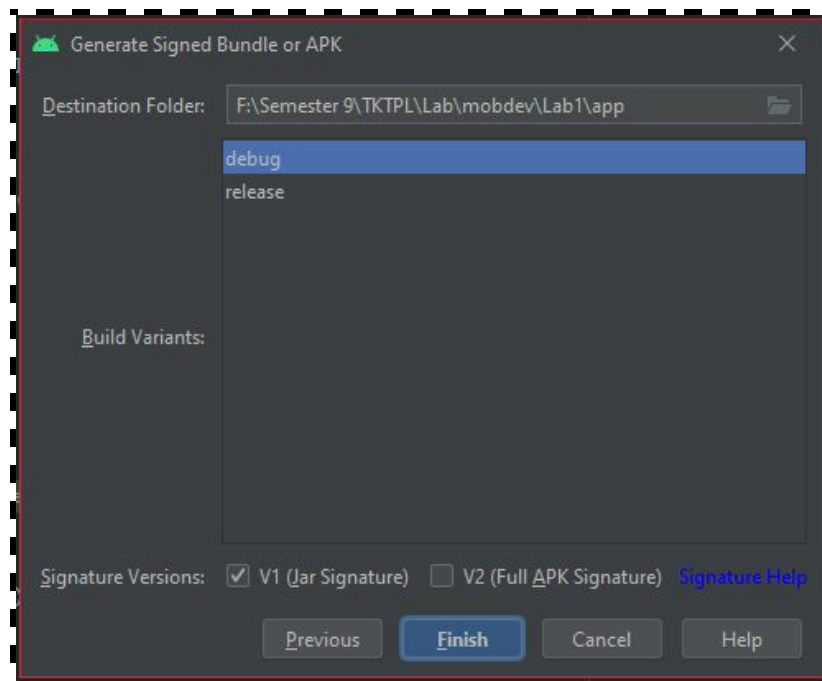
Akan muncul *window* seperti berikut. Terdapat keystore yang perlu diisi dan informasi identitas. Isi berdasarkan petunjuk pada link berikut <https://developer.android.com/studio/publish/app-signing#generate-key>. Hasilnya akan seperti berikut. Jika sudah, klik **OK**.



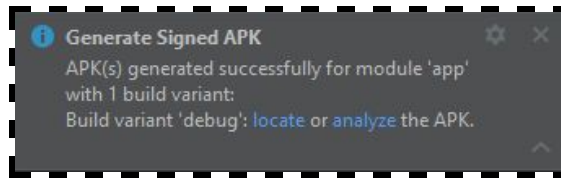
Sekarang kolom-kolom kosong tadi sudah terisi. Selanjutnya klik **Next**.



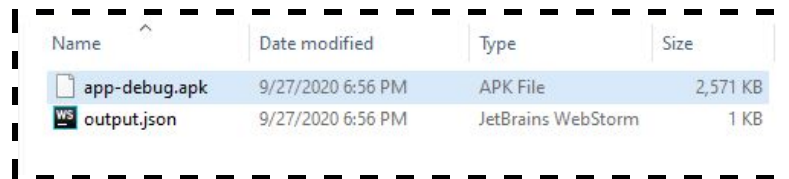
Disini kita akan memilih **variants** dari aplikasi dan **signature versions**. Pada bagian **variants**, saya memilih **debug** karena aplikasi ini belum selesai (*development version*) dan belum bisa disebut produk *beta version* apalagi *release version*. Pada bagian **signature versions**, saya memilih **V1 saja**, walaupun saya masih belum memahami kegunaan dari **signature versions** ini, saya setidaknya memilih salah satu saja. Klik **Finish**.



Setelah android selesai melakukan *built*, kita dapat lihat di bagian pojok kanan bawah terdapat notifikasi seperti berikut. Klik **locate** untuk membuka folder yang berisi *file* APK kita.

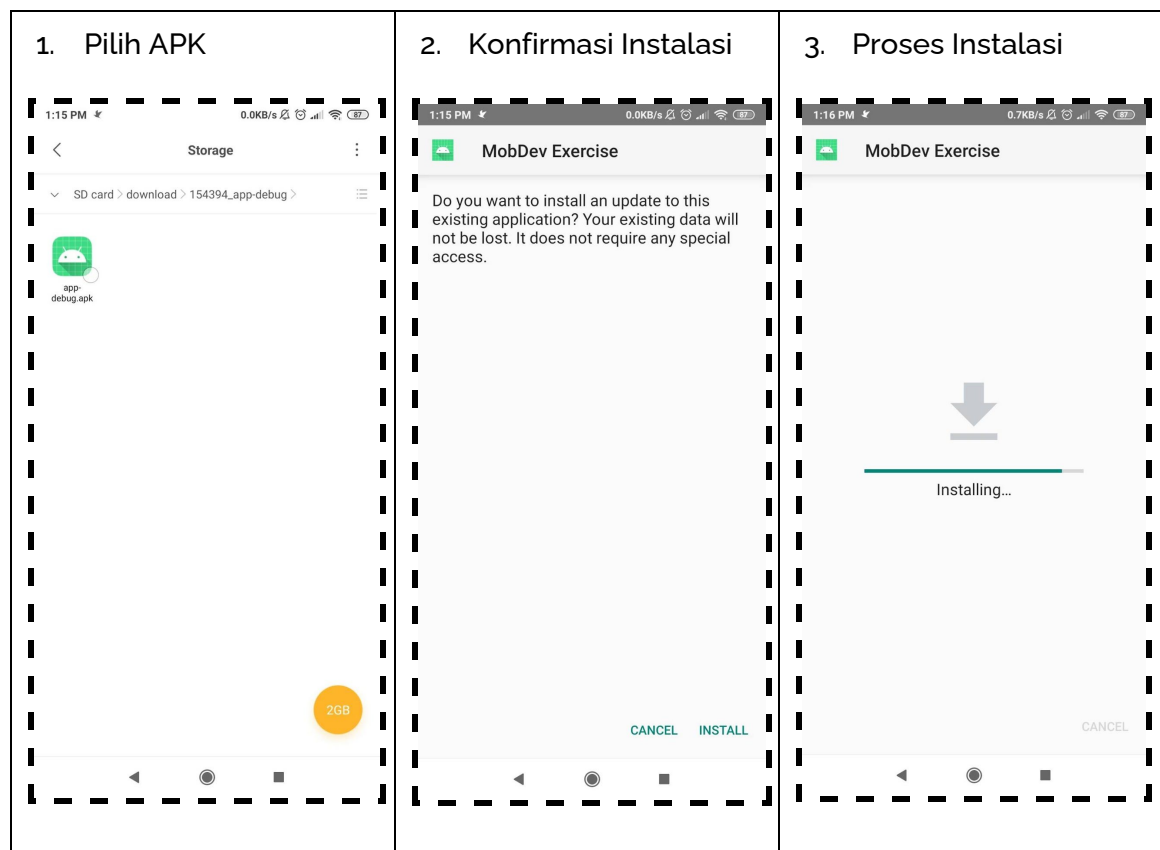


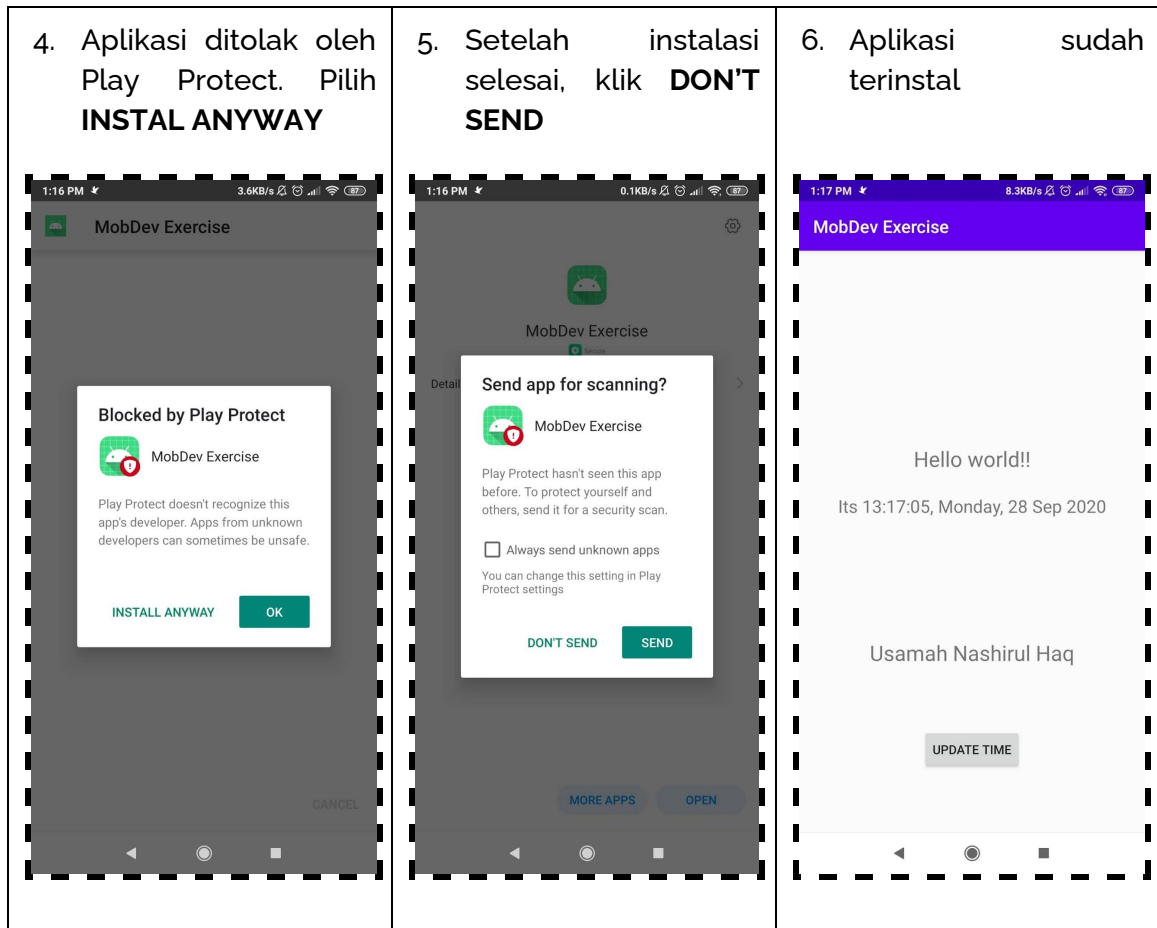
Kita bisa melihat *file* dengan nama **app-debug.apk** yang merupakan file APK yang telah di-*build* oleh android studio. *File* ini dapat ditransfer ke *smartphone* berbasis android untuk diinstal.



4. Install APK on Physical Device

Setelah saya mengirim *file* APK yang telah dibuat di tahap sebelumnya, saya melakukan instalasi pada *smartphone* saya. Berikut prosesnya.





Lesson learned

Pada tahap 4, muncul notifikasi *Blocked by Play Protect*. Notifikasi ini muncul dikarenakan aplikasi yang sudah saya buat belum terdaftar secara resmi di Play Store. *Smartphone* yang berbasis android akan mencoba mencegah proses instalasi dari aplikasi tidak dikenal agar terhindar dari masalah yang tidak diinginkan. Namun, jika seorang pengguna yakin dengan apa yang dia lakukan dan tahu aplikasi apa yang dia instal, pengguna juga dapat tetap menginstal aplikasi tersebut dengan memilih opsi **INSTAL ANYWAY** pada saat proses instalasi.

C. Kolaborator

- Bagus Pribadi - 1706043941
- Rahmadian Tio Pratama - 1706044074
- Steffi Alexandra - 1706043992
- Stefan Mayer Sianturi - 1606918364