



UNIVERSITAS SEMARANG
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
TEKNIK INFORMATIKA

Mobile Programming

Modul Praktikum Mahasiswa

Oleh:

Alauddin Maulana Hirzan, S. Kom., M. Kom
NIDN. 0607069401

Daftar Isi

Pendahuluan	2
0.1 Mengenal Android	2
0.2 Mengenal Android Studio	2
0.3 Mengenal Android SDK	3
0.4 Mengenal Android Emulator	3
Persiapan Praktikum	5
0.5 Perangkat Keras	5
0.6 Perangkat Lunak	5
1 Praktikum 1	6
1.1 Konfigurasi Android Studio dan Android Emulator	6
1.2 Tutorial	6
2 Praktikum 2	16
2.1 Antarmuka Halaman Login	16
2.2 Tutorial	16
3 Praktikum 3	27
3.1 Pemrograman Halaman Login	27
3.2 Tutorial	27
4 Praktikum 4	35
4.1 Pemrograman Halaman Pendaftaran	35
4.2 Tutorial	35
5 Praktikum 5	43
5.1 SQLite Insert Data	43
5.2 Tutorial	43
6 Praktikum 6	50
6.1 SQLite Read Data	50
6.2 Tutorial	50
7 Praktikum 7	59
7.1 SQLite Update Data	59
7.2 Tutorial	59

8	Praktikum 8	69
8.1	Google Map	69
8.2	Tutorial	69

Daftar Gambar

1	Perangkat Android	2
2	Android Studio	3
3	Android SDK	3
4	Android Emulator	4

Pendahuluan

0.1 Mengetahui Android

Sistem operasi Android adalah sistem operasi seluler untuk digunakan terutama untuk perangkat layar sentuh, ponsel, dan tablet. Desainnya memungkinkan pengguna memanipulasi perangkat seluler secara intuitif, dengan gerakan jari yang mencerminkan gerakan umum, seperti mencubit, menggesek, dan mengetuk.



Gambar 1: Perangkat Android

0.2 Mengetahui Android Studio

Sebuah *Integrated Development Environment* (IDE) yang digunakan untuk mengembangkan aplikasi Android. IDE ini merupakan IDE yang diperuntukkan untuk pemula karena sederhana dan tidak perlu menambahkan hal-hal kompleks secara manual untuk aplikasi sederhana



Gambar 2: Android Studio

0.3 Mengetahui Android SDK

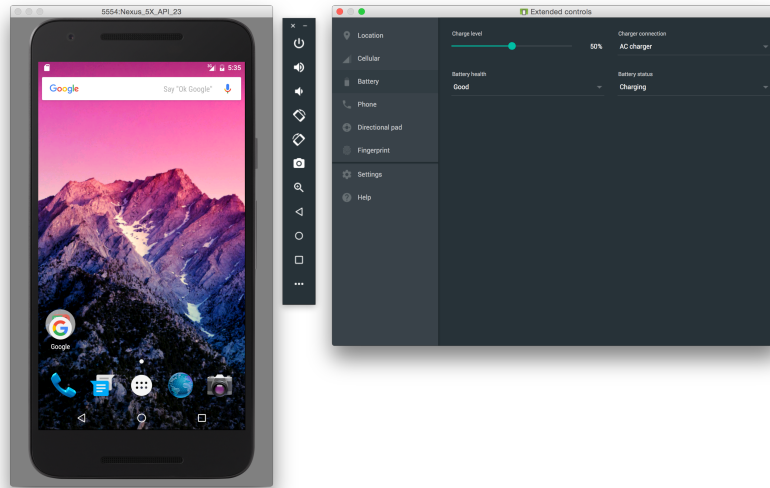
Merupakan framework dasar untuk membuat aplikasi Android dan sudah tersedia secara default di Android Studio. Android SDK memiliki berbagai macam versi tergantung dari versi Android itu sendiri. Penggunaan Android SDK juga akan mempengaruhi kompatibilitas ke belakang dengan Android versi lama.



Gambar 3: Android SDK

0.4 Mengetahui Android Emulator

Android Studio juga menyediakan Android Emulator yang digunakan untuk menghubungkan perangkat Android Virtual ke IDE. Meskipun dianggap cukup berat, namun Emulator ini dapat membantu proses pembuatan aplikasi Android bagi mereka yang tidak memiliki perangkat Android



Gambar 4: Android Emulator

Persiapan Praktikum

Agar praktikum dapat berjalan dengan lancar, mahasiswa diwajibkan memenuhi persyaratan berikut baik dalam bentuk perangkat keras maupun lunak:

0.5 Perangkat Keras

- Prosesor dengan 4 inti
- RAM minimal 4GB, rekomendasi 8GB
- HDD 10GB

0.6 Perangkat Lunak

Perangkat lunak berikut ini wajib diinstall oleh mahasiswa demi lancarnya praktikum:

- Android Studio terbaru
 - Download via <https://developer.android.com/studio>
- Universal ADB Driver
 - Download via <https://adb.clockworkmod.com/>

Bab 1

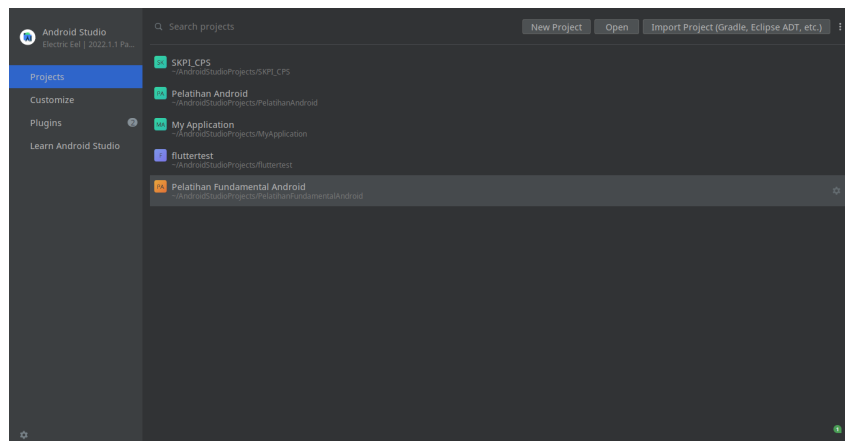
Praktikum 1

1.1 Konfigurasi Android Studio dan Android Emulator

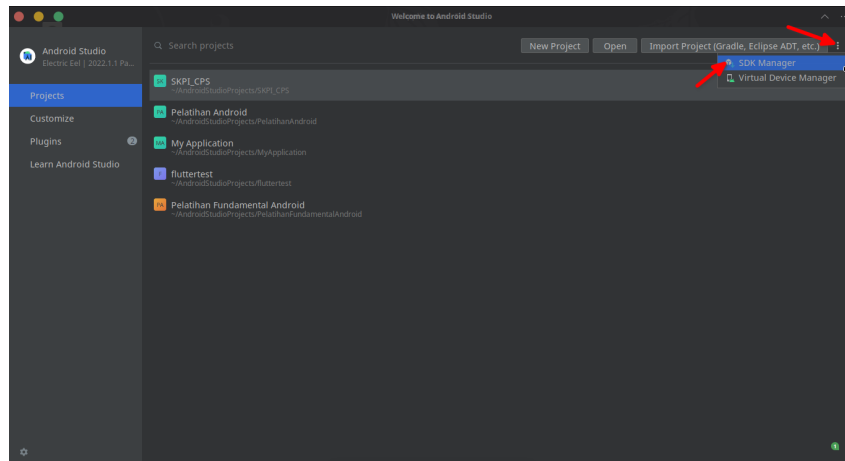
Di bagian ini mahasiswa diajarkan bagaimana mengkonfigurasi Android Studio khususnya pada bagian Android SDK, dan Android Emulator di komputer masing-masing. Hal ini sangat direkomendasikan agar mahasiswa dapat mengenali struktur Android Studio dengan lebih mudah,

1.2 Tutorial

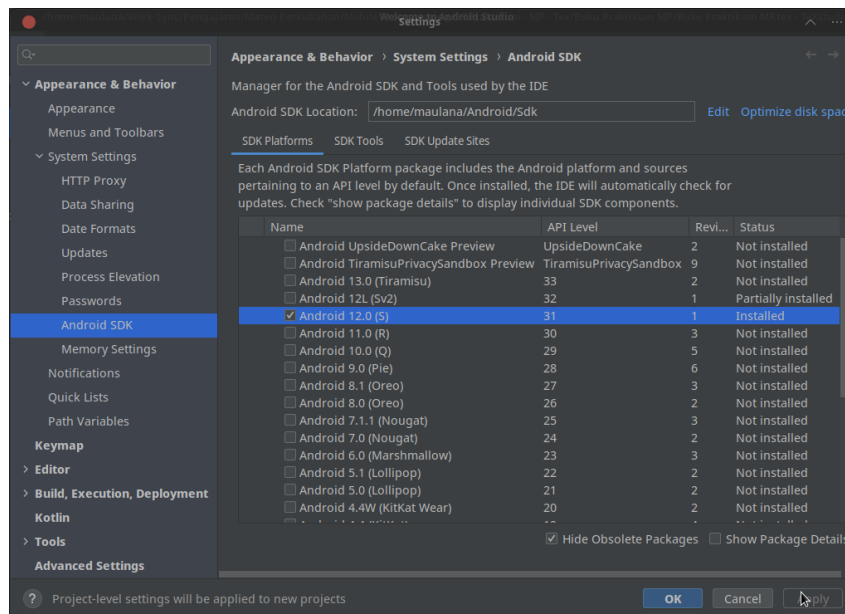
1. Buka **Android Studio**, dan pastikan menggunakan **Versi Terbaru** untuk memulai praktikum ini.



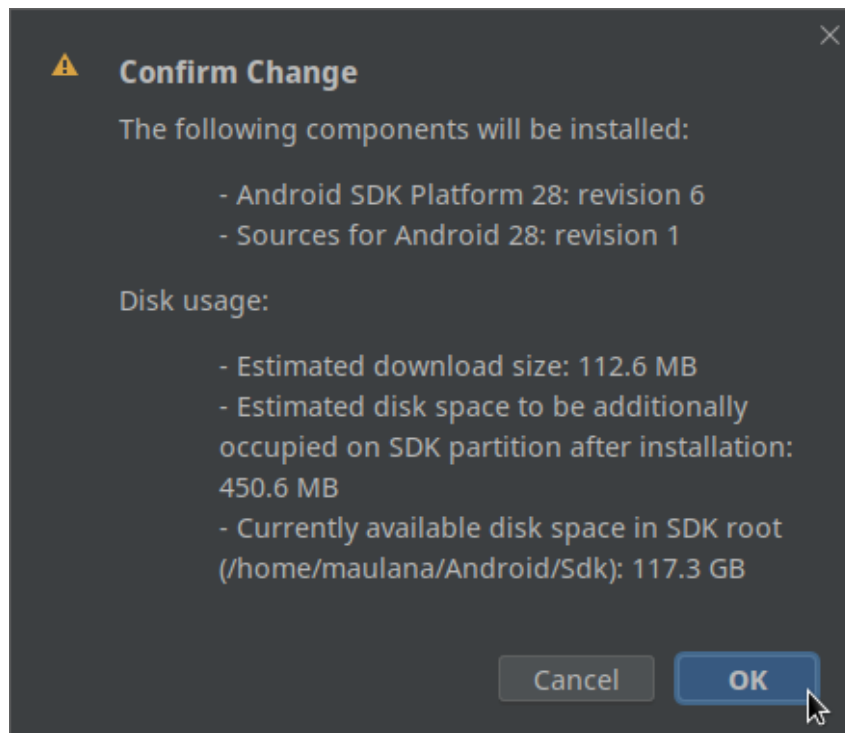
2. Untuk mengkonfigurasi Android SDK, klik **Titik Tiga / More Action** → **SDK Manager**



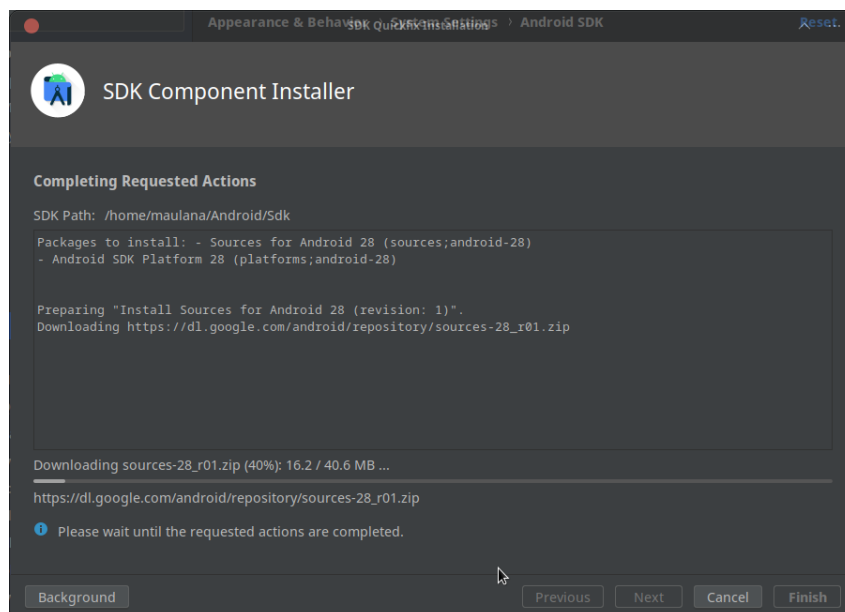
3. Jika sudah muncul tampilan **SDK Manager**, cek apakah ada SDK yang sudah terinstall. Secara **Default**, SDK sudah terpasang secara otomatis bersamaan dengan instalasi Android SDK.



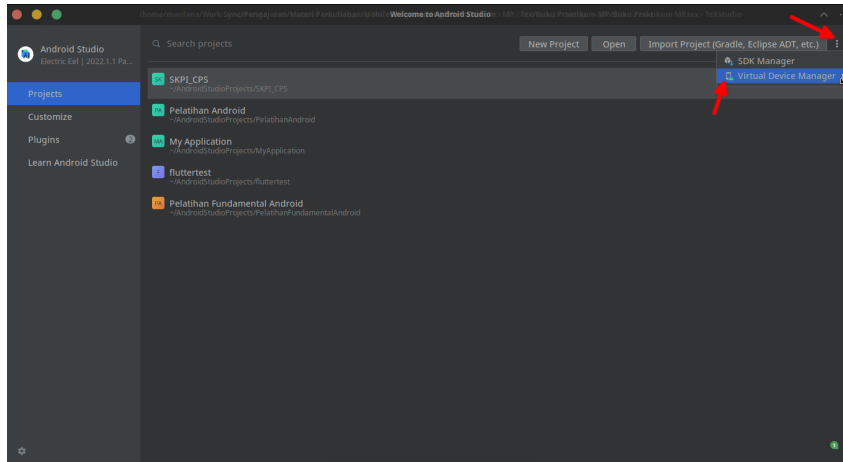
4. Untuk menginstall SDK secara manual, cukup centang SDK target dan klik **Apply**. Android Studio akan memunculkan peringatan (jika ada, klik OK)



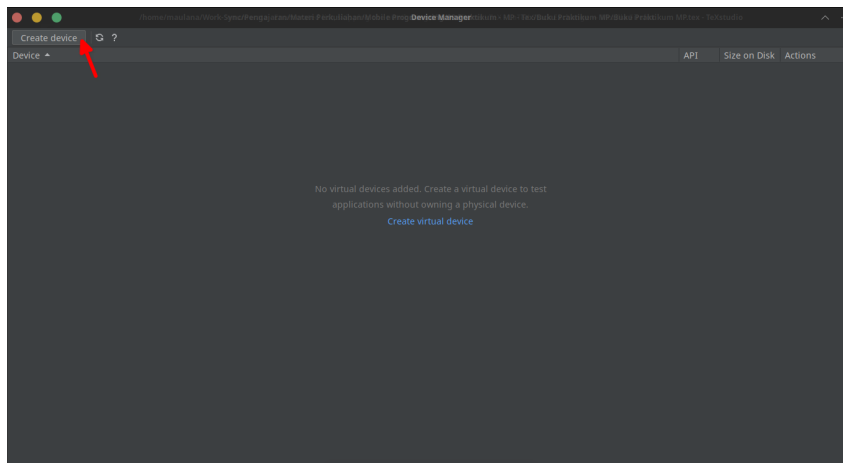
5. Kemudian proses instalasi SDK akan dimulai. Tunggu hingga selesai dan klik **Finish** untuk menutup **SDK Component Installer**



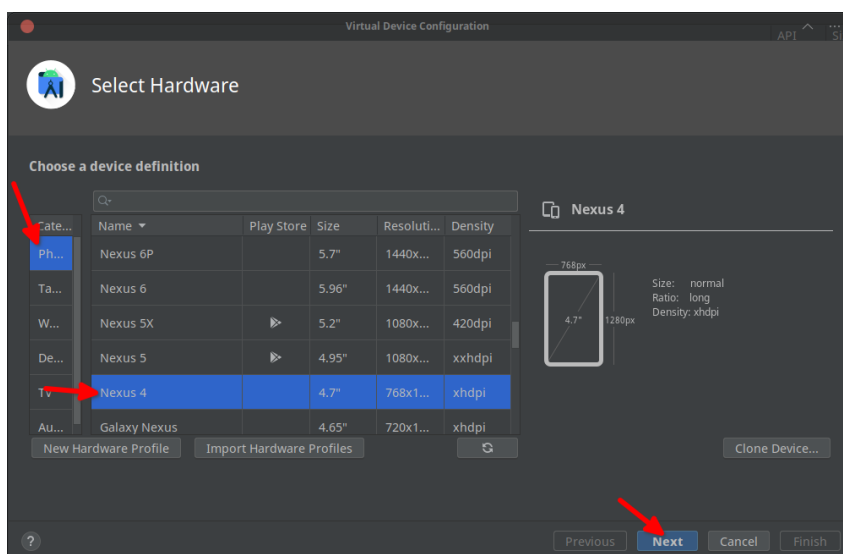
6. Lalu klik **OK** untuk menutup **SDK Manager**
7. Android Studio akan kembali menampilkan halaman **Welcome to Android Studio**. Kemudian konfigurasi perangkat virtualisasi dengan klik **Titik Tiga / More Action** → **Virtual Device Manager**



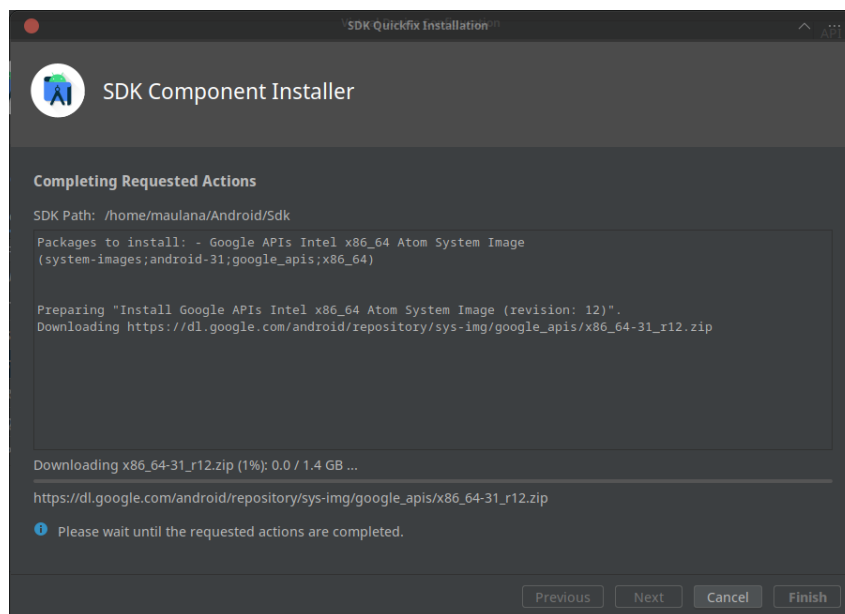
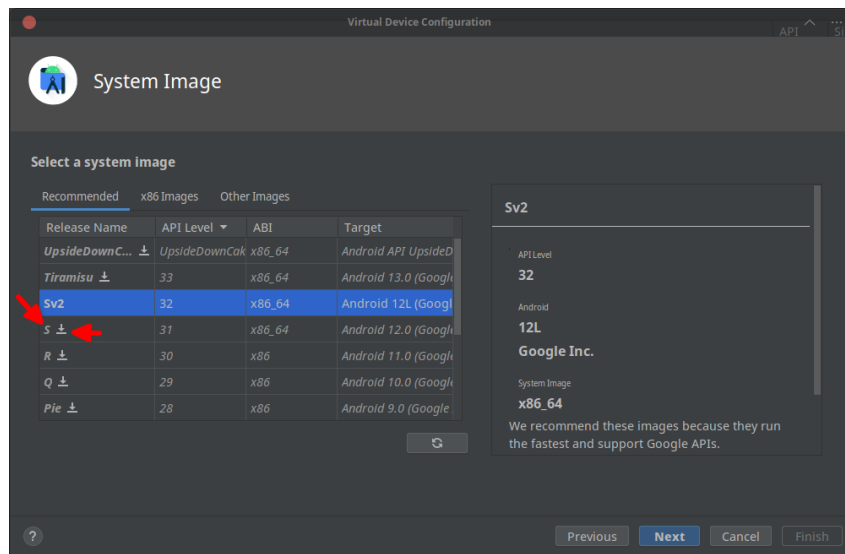
8. Berikutnya akan memunculkan konfigurasi perangkat Virtual. Secara **Default** halaman ini masih kosong, dan bisa memulai membuat dengan meng-klik **Create device**



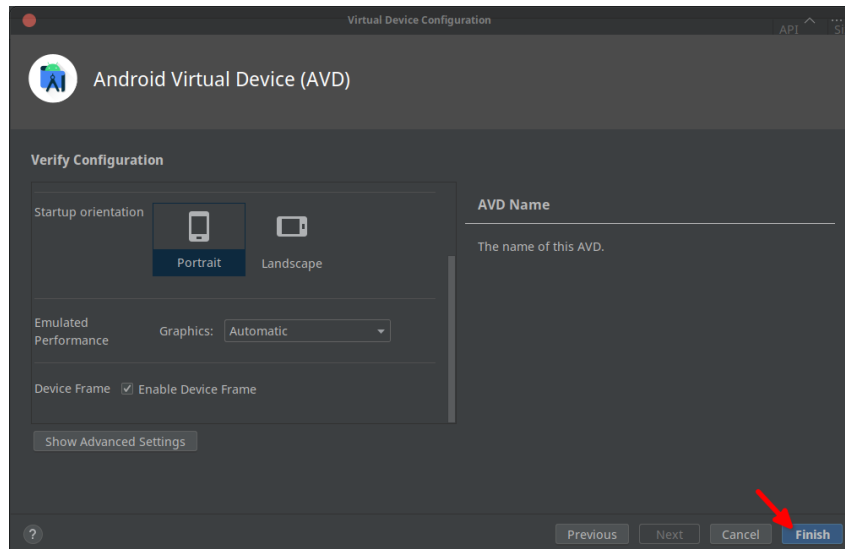
9. Android Studio akan memulai *Wizard* untuk membuat perangkat **Virtual Android**. Pilih **Phone** → **Nexus 4** → **Next**



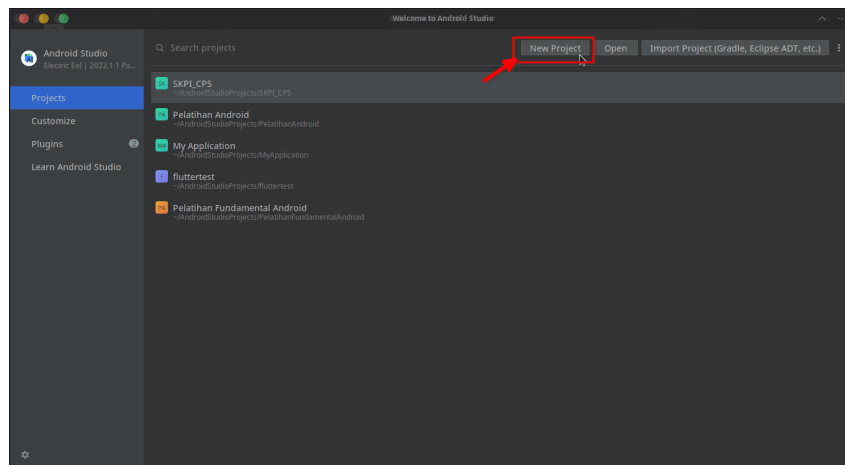
10. Pilih **Versi Android**, → Klik **Icon Download** di samping versi apabila belum terinstall → Klik **Next** apabila **System Image** sudah terinstall.



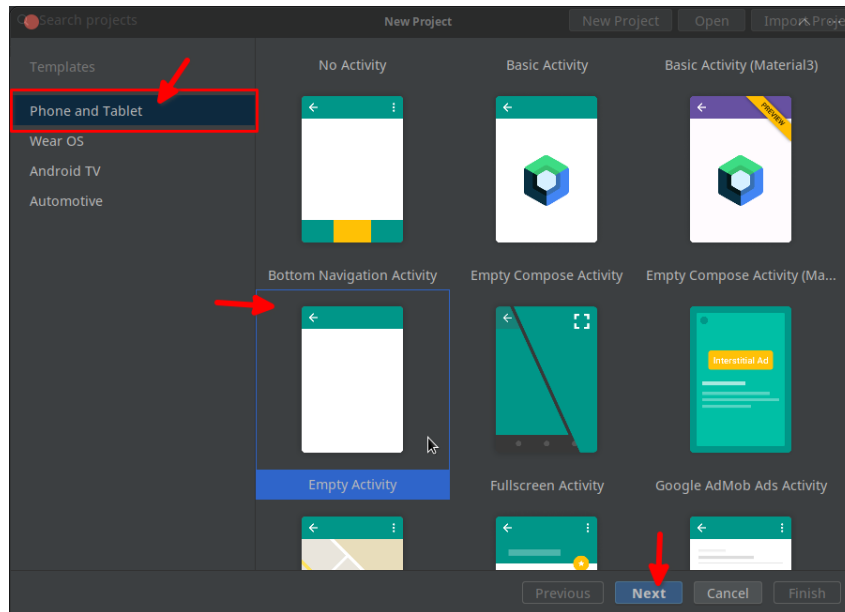
11. Berikutnya adalah konfigurasi perangkat itu sendiri, secara **Default** tidak perlu dirubah kecuali memahami konfigurasi lanjutan dari **Virtual Machine**. Klik **Finish**



12. Jika **Android Studio** meminta untuk menginstall **Intel HAXM**, instal aplikasi tersebut untuk dapat menggunakan **emulator**. Dan pastikan BIOS/UEFI komputer sudah terkonfigurasi untuk **Virtualisasi**
13. Tutup **Virtual Device Manager**, dan Android Studio sudah siap untuk membuat aplikasi. Klik **New Project**.

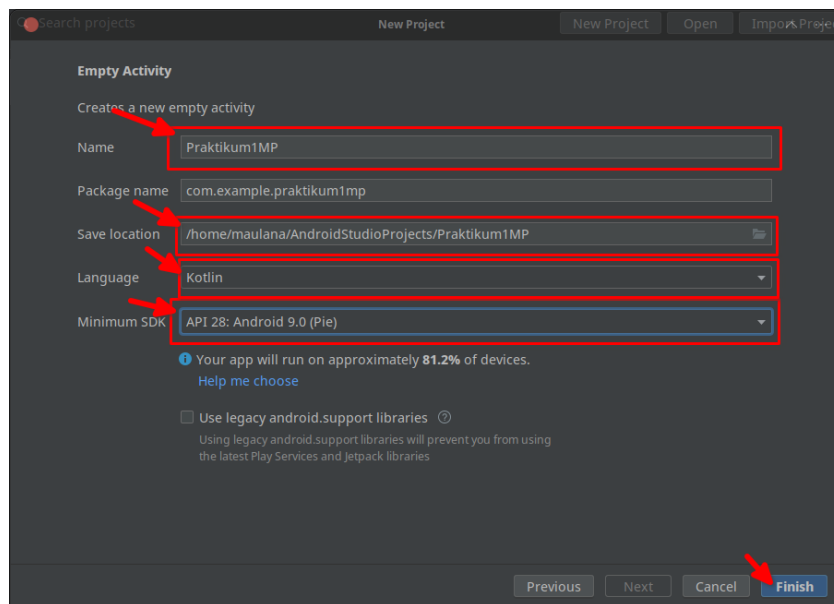


14. Android Studio akan membuka memperlihatkan konfigurasi proyek yang akan dibuat. Pilih **Phone and Tablet** → Klik **Empty Activity** => Klik **Next**

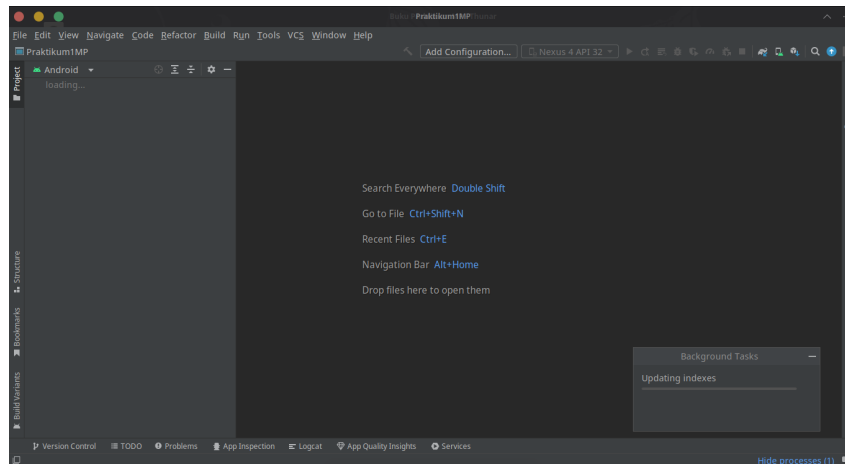


15. Sesuaikan konfigurasi proyek seperti berikut, dan klik **Finish**:

- Nama Projek : `Praktikum1MP`
- Lokasi Projek : `<Sesuaikan Folder Mahasiswa>`
- Bahasa : Kotlin
- Minimum SDK : `<Sesuaikan Perangkatl Minimal>`

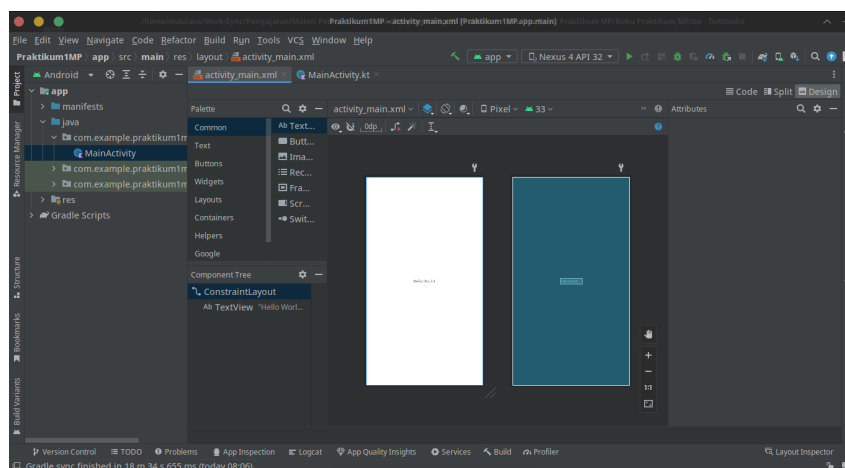
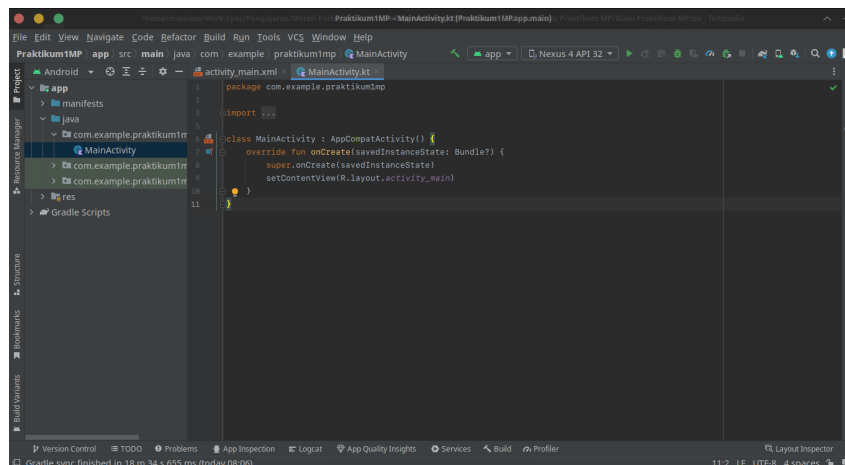


16. Android Studio akan menyiapkan file proyek, tunggu hingga selesai. Android Studio akan memperlihatkan kode-kode dan **Project Tree** di samping.

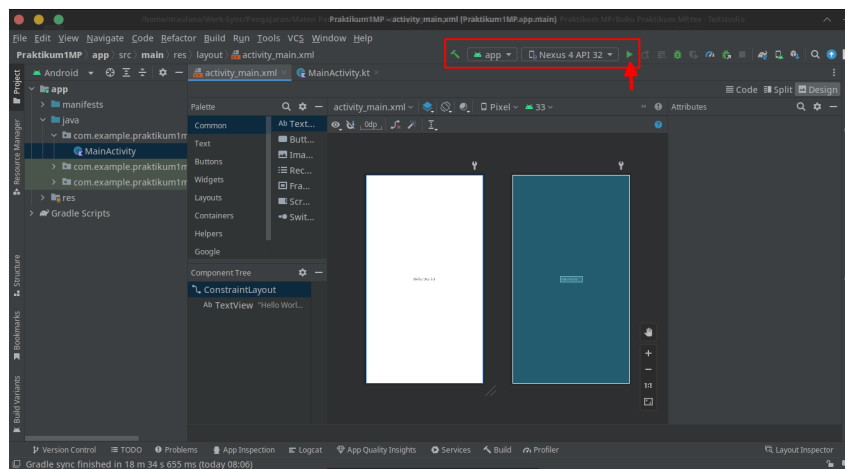


17. Android Studio akan membuat dua (2) file secara **default**:

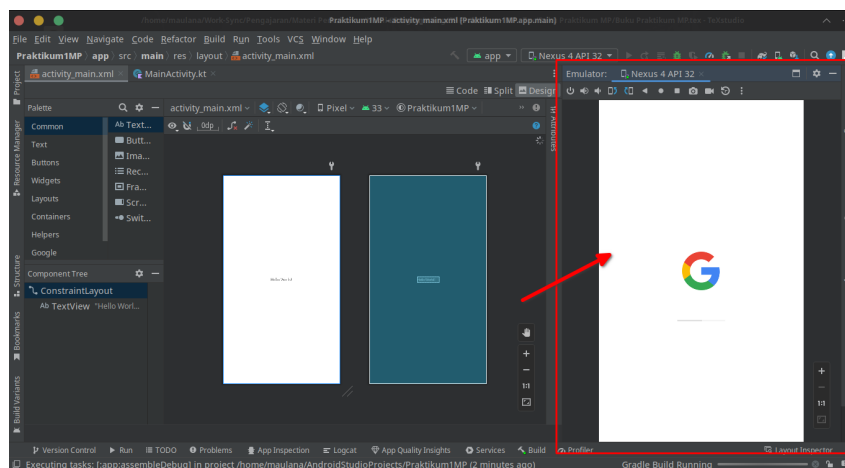
- **MainActivity.kt** : Digunakan untuk menaruh kode-kode fungsional menggunakan bahasa **Java/Kotlin**
- **activity_main.xml** : Digunakan untuk meletakkan objek-objek antarmuka aplikasi Android



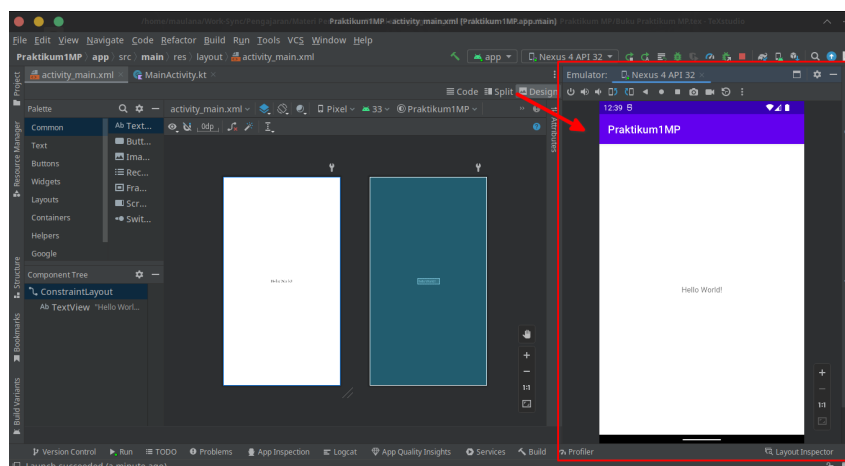
18. Android Studio sudah siap untuk membuat aplikasi Android. Namun sebelum membuat aplikasi, berikutnya adalah mencoba untuk menjalankan aplikasi **Hello World** ini ke **Virtual Device** dengan klik tombol **Play**



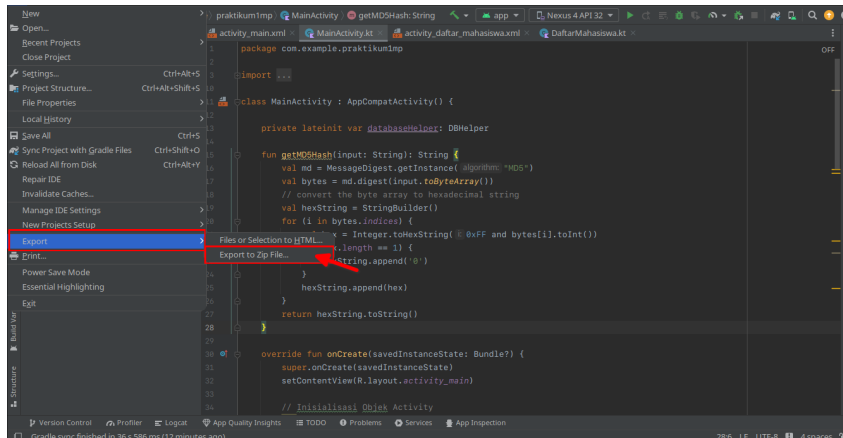
19. Tergantung dari versi Android Studio, versi terbarunya akan membuka **Panel Kanan** untuk menampilkan **Virtual Android**. Tunggu **Kompilasi** dan **Booting** perangkat hingga aplikasi muncul di layar tersebut



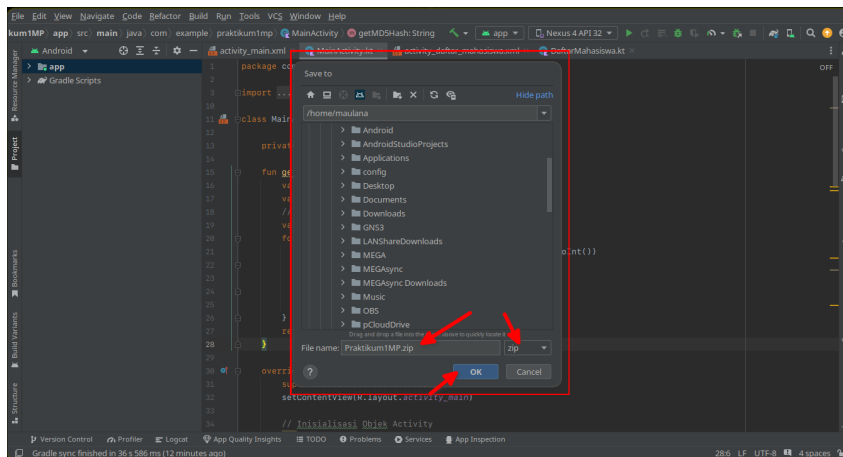
20. Jika sukses, maka Android Studio akan menampilkan aplikasi seperti gambar berikut



21. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



22. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**



Bab 2

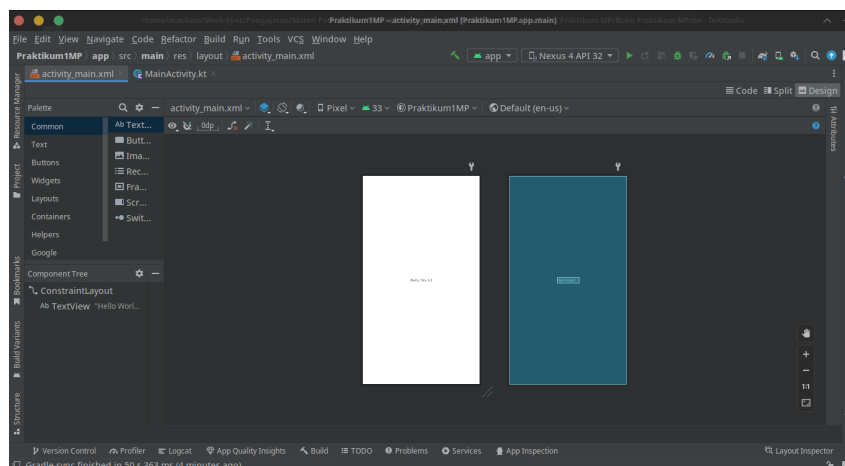
Praktikum 2

2.1 Antarmuka Halaman Login

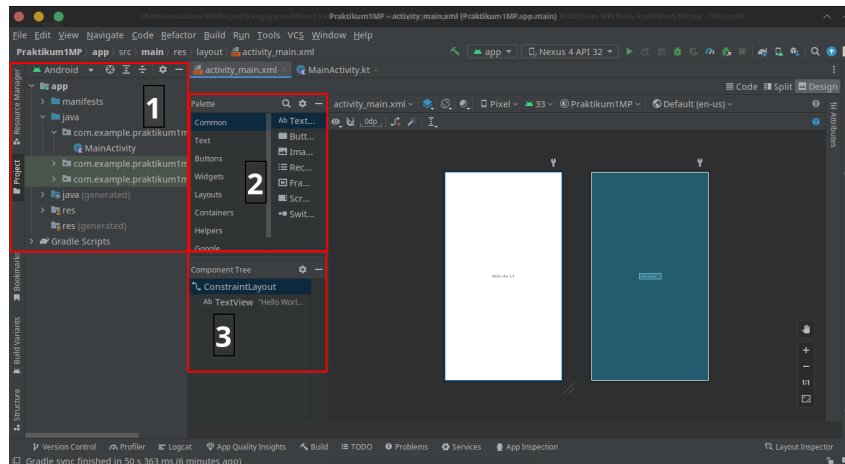
Di bagian ini mahasiswa akan diperkenalkan dengan antarmuka **Android Studio**, sehingga dapat membantu siswa dalam melakukan navigasi yang ada di dalam aplikasi tersebut.

2.2 Tutorial

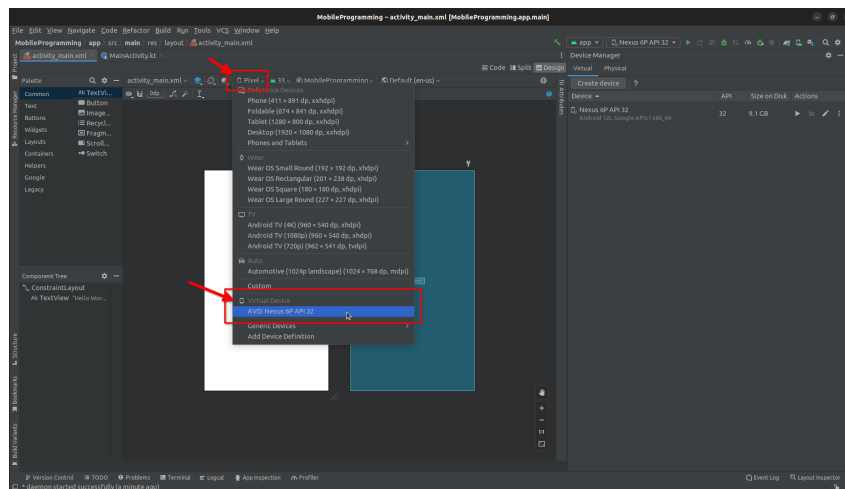
1. Buka **Android Studio**, dan buka kembali **Praktikum1MP** yang sudah dibuat sesi sebelumnya.



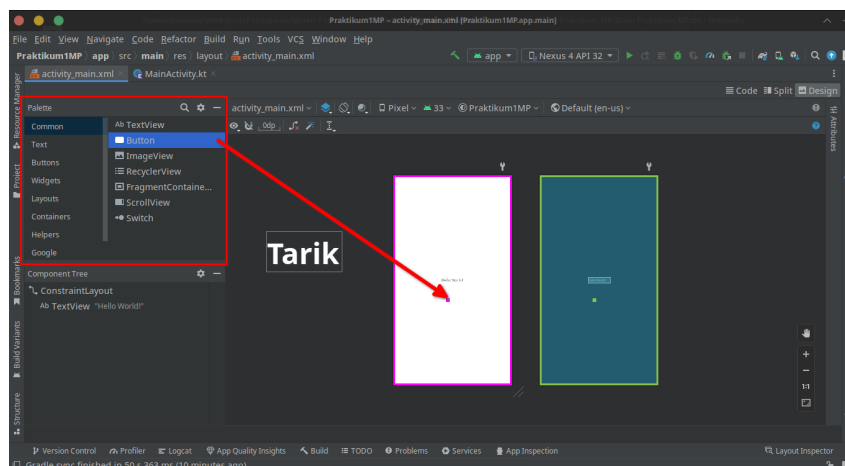
2. Di bagian panel kiri ada dua (2) hal yang tidak boleh dilupakan:
 - (a) Panel **Project** : Menampilkan projek saat ini (bisa dikecilkan)
 - (b) Panel **Palette** : Menampilkan objek yang bisa ditempel
 - (c) Panel **Component Tree** : Menampilkan objek yang sudah ditempel



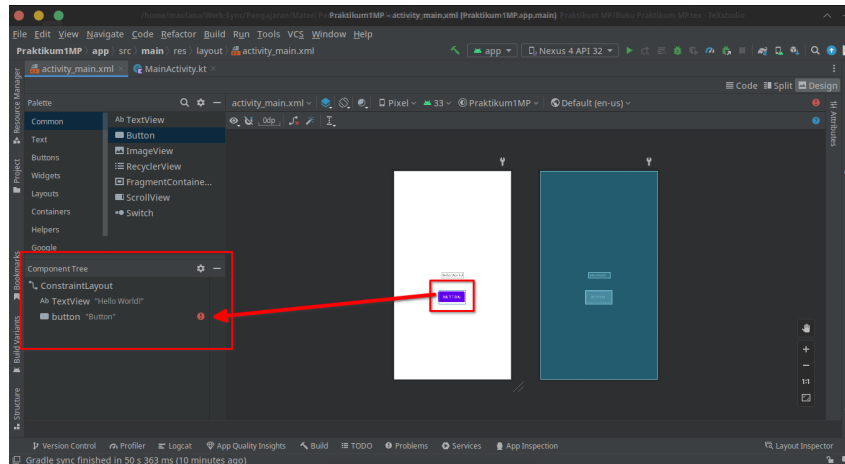
3. Agar **Activity** memiliki ukuran yang sesuai dengan ukuran perangkat, dapat dikonfigurasi melalui tulisan **Pixel** di atas **Activity**, lalu pilih **Virtual Device: AVD**. Activity akan merubah ukurannya secara otomatis



4. Untuk menempelkan objek dari **Palette**, cukup lakukan **Klik dan Tarik** objek ke **Kanvas Putih**. Perhatikan gambar untuk menambahkan Tombol ke **Kanvas**



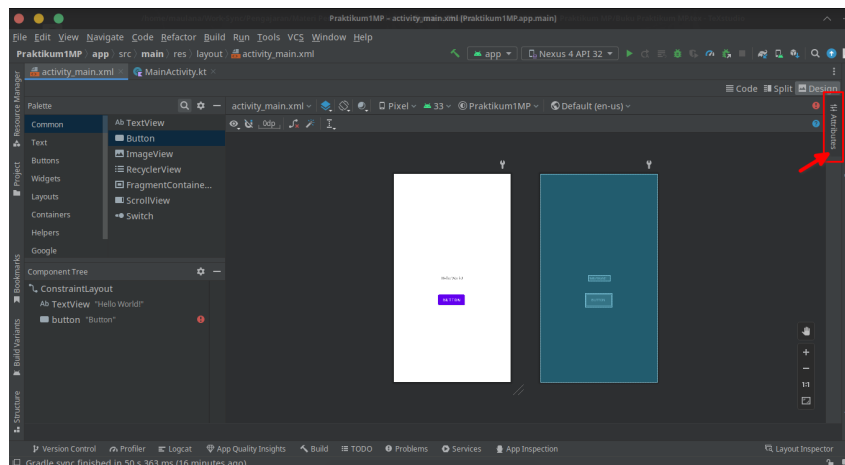
5. Jika dilakukan dengan benar, maka **Tombol** akan terlihat di **Kanvas Putih** dan **Component Tree** di Panel Kiri

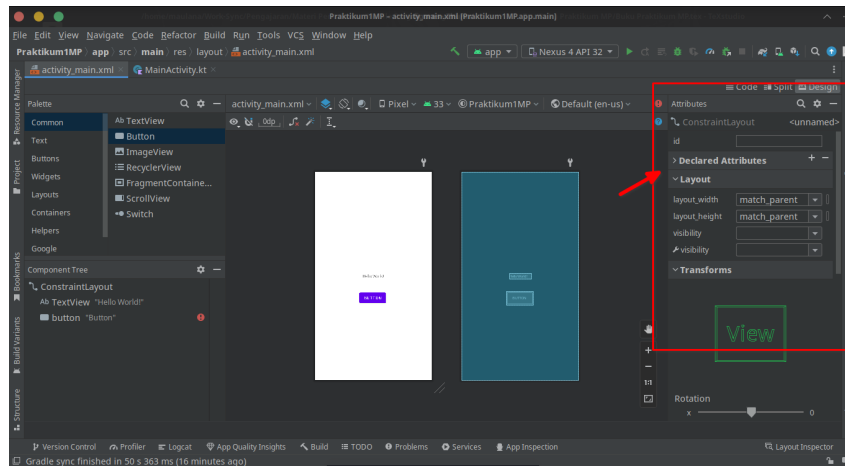


6. Namun objek ini masih memiliki *error* yang terlihat di **Component Tree**. Biasanya error ini muncul karena:

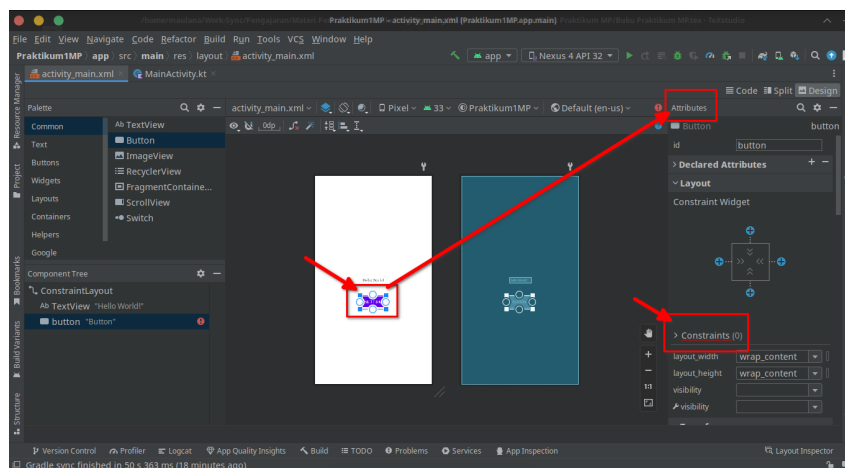
- (a) 1 **Constraint** Vertikal dan 1 **Constraint** Horizontal tidak diatur
- (b) Ukuran **Tombol** terlalu kecil

7. Karena objek baru ditambahkan, maka **Constraint** belum diatur, untuk mengatur ini klik di Panel Kanan tombol **Attributes**

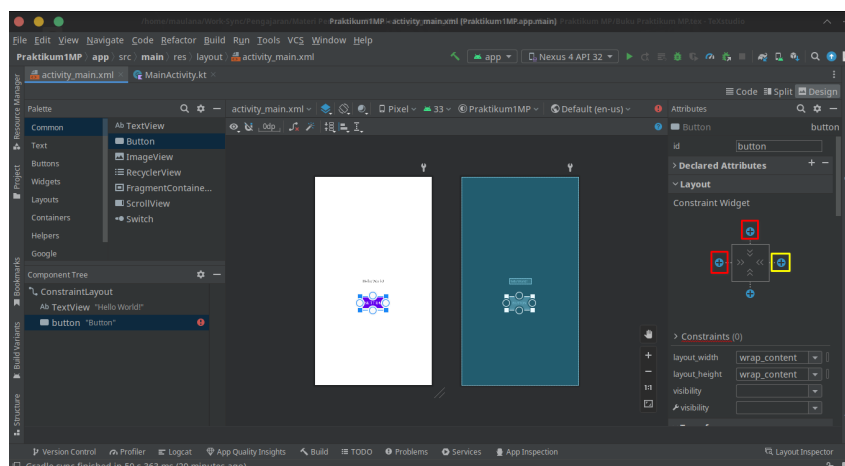




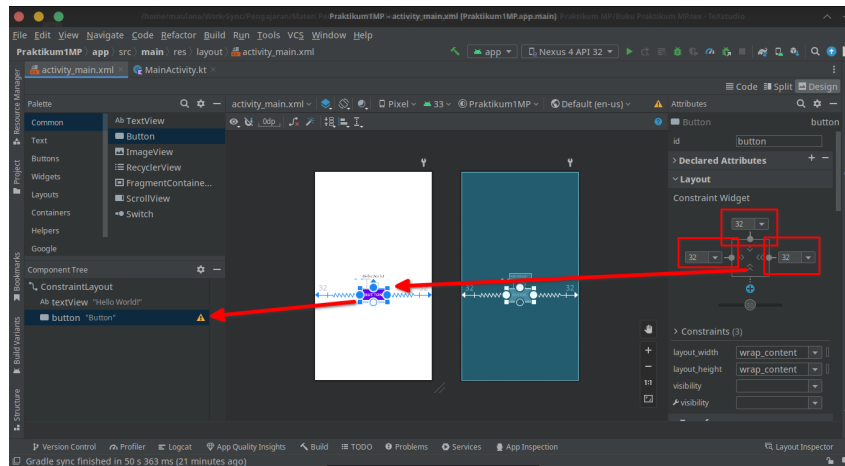
8. Untuk mengkonfigurasi **Attributes** dari suatu objek, pastikan panel **Attributes** terbuka, dan klik **Objek** yang mau dirubah. Berdasarkan gambar di bawah, **Constraint** tidak di set sehingga muncul indikator **Merah**



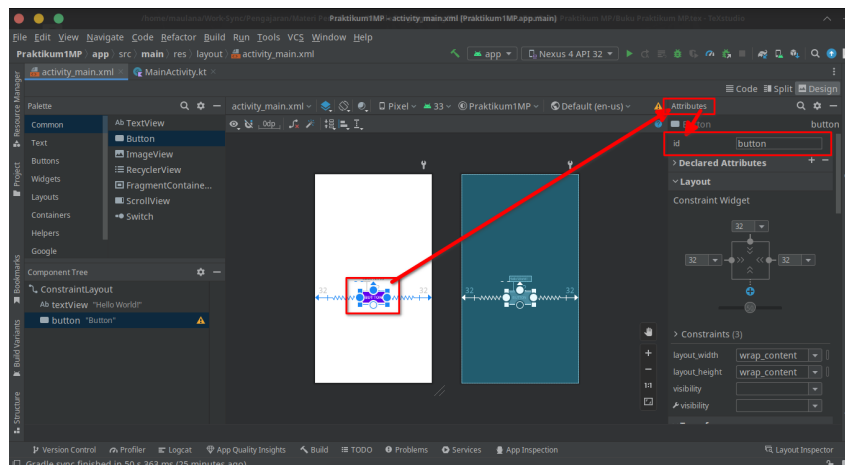
9. Untuk memperbaiki ini, cukup tambahkan **Constraint Atas** dan **Constraint Samping** seperti gambar berikut (Merah Wajib, Kuning Opsional):



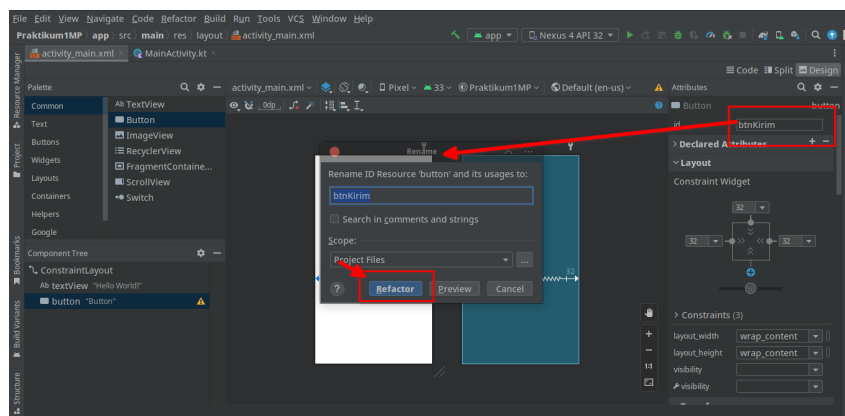
10. Atur jarak **Constraint** secara manual (Contoh 32) jika diperlukan, dan **error** pada objek **Tombol** sudah hilang.



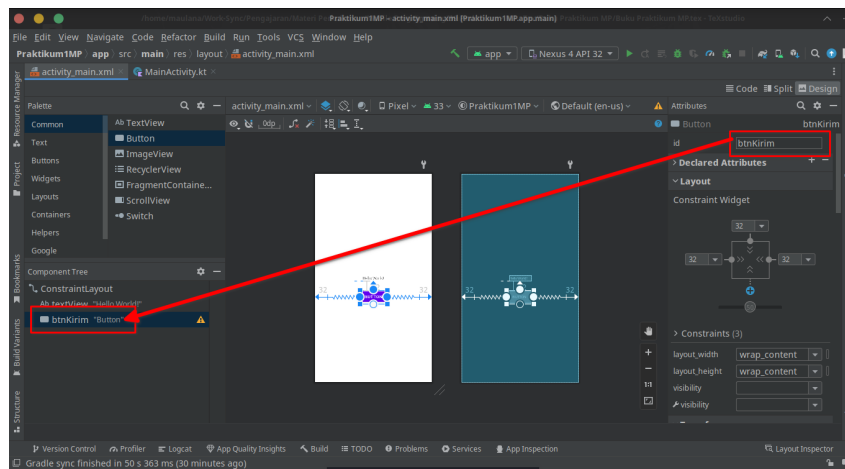
11. Dalam aplikasi Android memiliki banyak kemungkinan untuk ditambahkan objek dalam jumlah yang banyak. Maka untuk membedakan objek satu dengan yang lainnya dapat menggunakan **Attribute ID**
12. Untuk mengatur **ID Objek** cukup klik **Objek** dan lihat panel **Attribute**. Naikkan panel sampai paling atas.



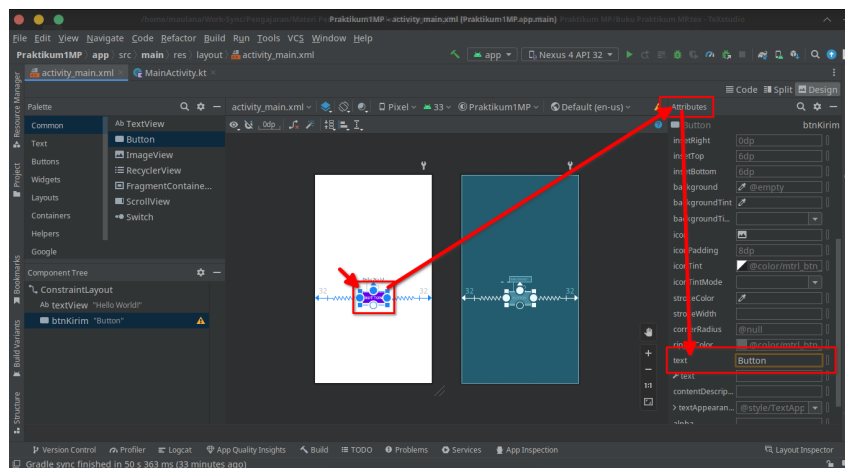
13. Ubah **ID Objek** menjadi **btnKirim** (atau nama unik lainnya). Klik area bebas maka permintaan **Rename** akan muncul. Klik **Refactor** untuk menerapkan penamaan ID Objek.



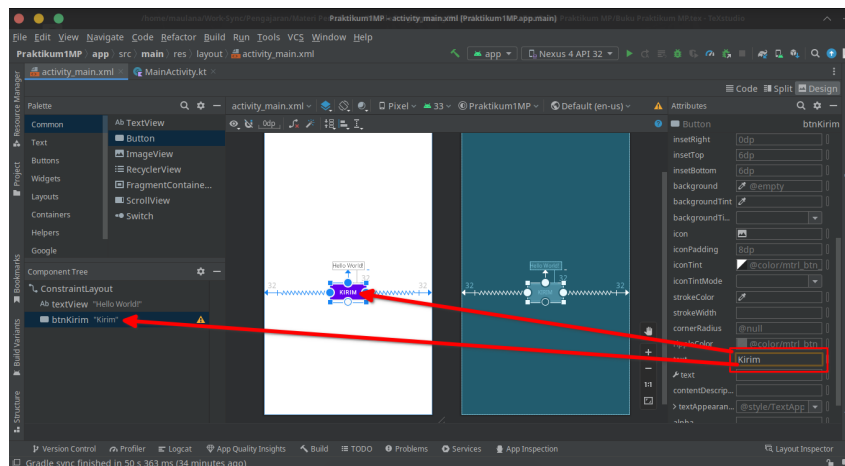
14. Penamaan ini akan langsung berdampak ke **Component Tree**. Maka untuk mengakses **Tombol** ini cukup memanggil **ID Objek** tersebut



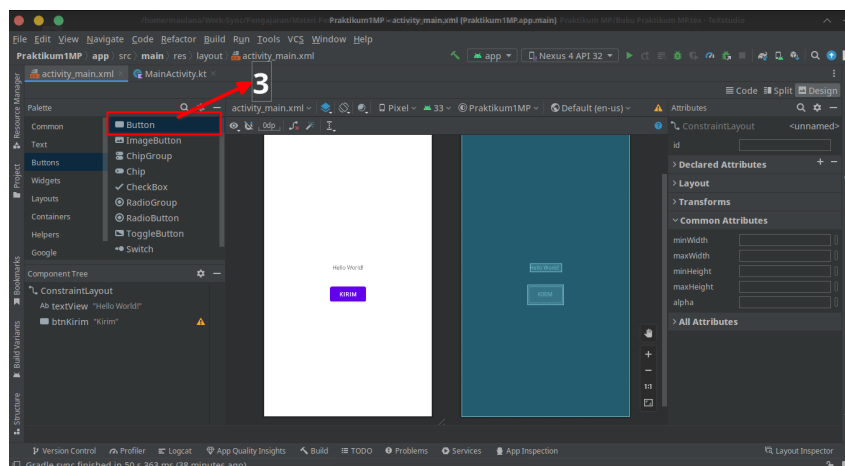
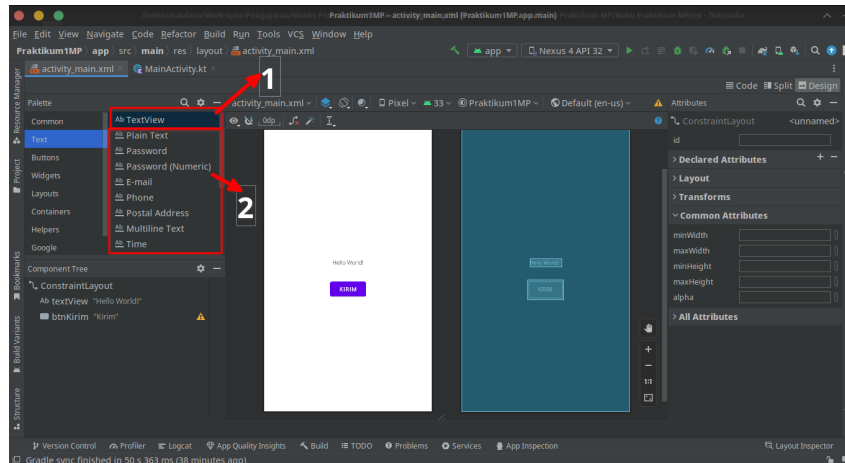
15. Berikutnya adalah mengubah **Isi Teks**. Dengan cara yang sama namun ketika memilih panel **Attribute**, pilih **Text** (Gunakan **Hint** untuk **EditText**)



16. Isi langsung dengan teks yang diinginkan lalu **Enter**

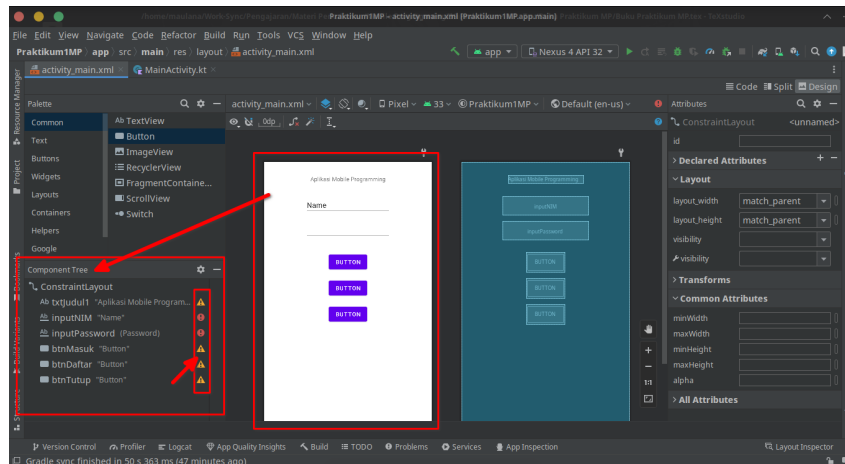


17. Konfigurasi ini berlaku untuk **Semua Objek**. Sehingga memudahkan dalam konfigurasinya.
18. Selain itu ada beberapa objek lain yang wajib diketahui berupa:
 - (a) **TextView** : Digunakan untuk menampilkan teks / String saja tanpa input
 - (b) **EditText** : Digunakan untuk menampilkan dan menerima input dalam bentuk teks / String
 - (c) **Button** : Digunakan untuk pemberi aksi ketika ditekan.

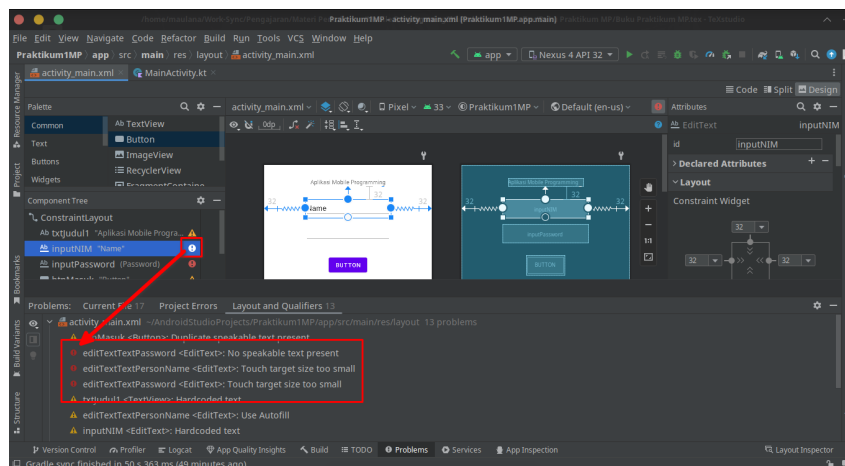


19. Hapus semua objek yang ada di **Kanvas Putih**. Lalu buatlah tampilan seperti berikut dengan menggunakan:

- Susunlah satu per satu dari atas ke bawah
 - 1 **TextView**
 - 1 **EditText (Plain Text)**
 - 1 **EditText (Password)**
 - 3 **Button**
- Atur **Constraint SEMUA OBJEK** di atas seperti Langkah 9 dan 10

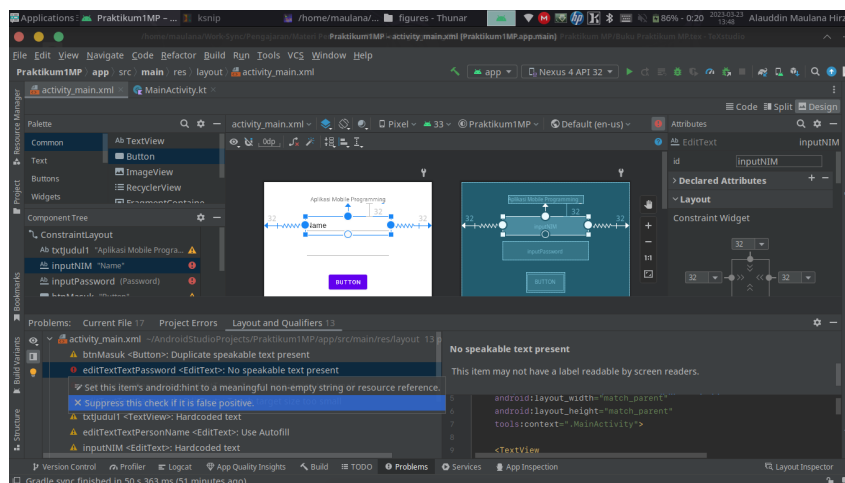


20. Ketika semua terpasang, maka akan muncul khususnya untuk **EditText** nya. Klik **Indikator Error Merah** untuk menampilkan error.

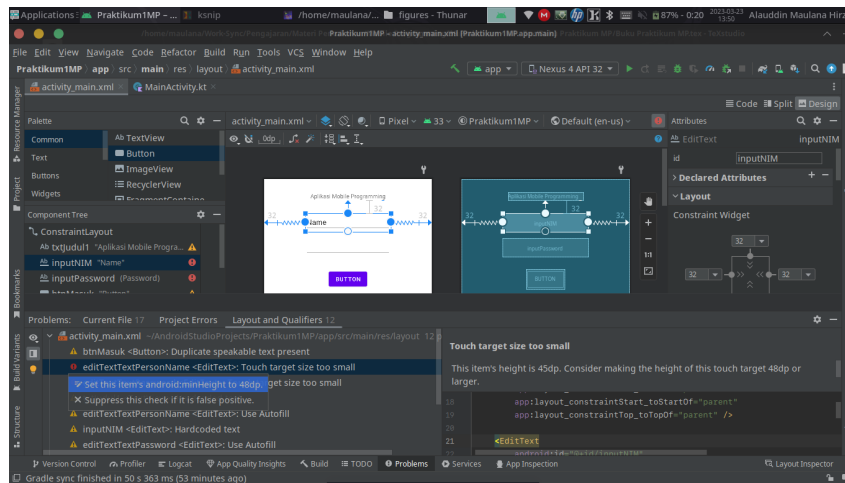


21. Untuk memperbaiki ini, gunakan **Klik Kanan**, pilih **Show Quick Fixes** lalu pilih:

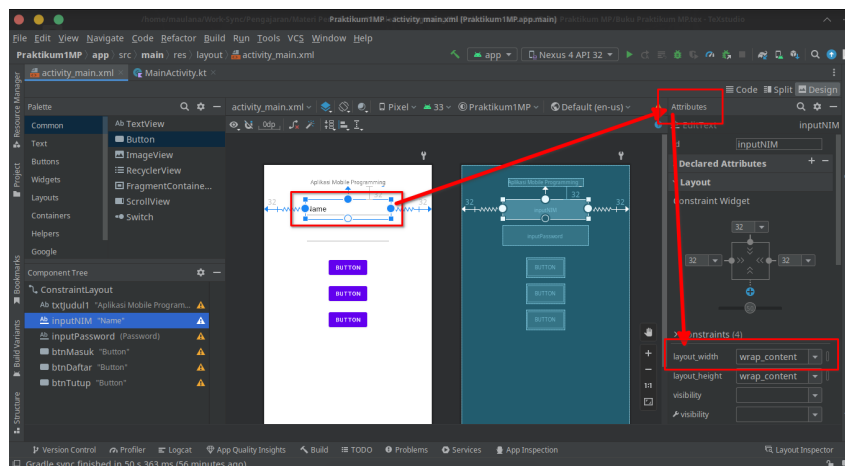
- Jika **Error No speakable text present** → **Suppress this check if it is false positive**



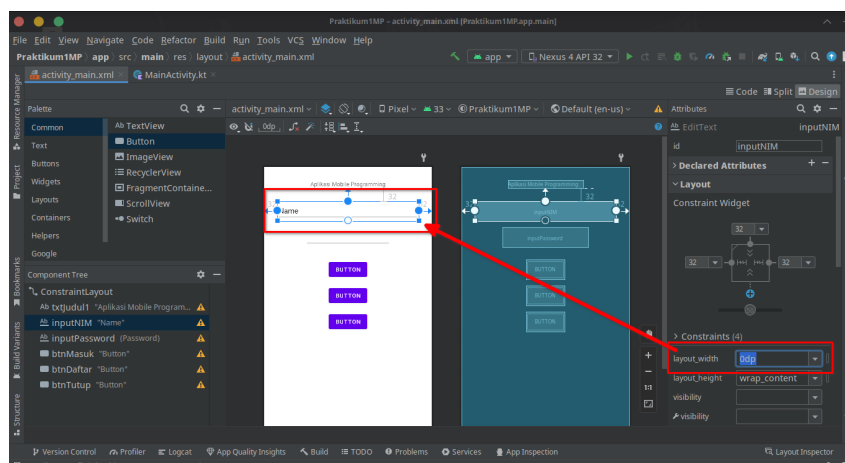
- Jika Error Touch target size too small → Set this item's android:minHeight to 48dp



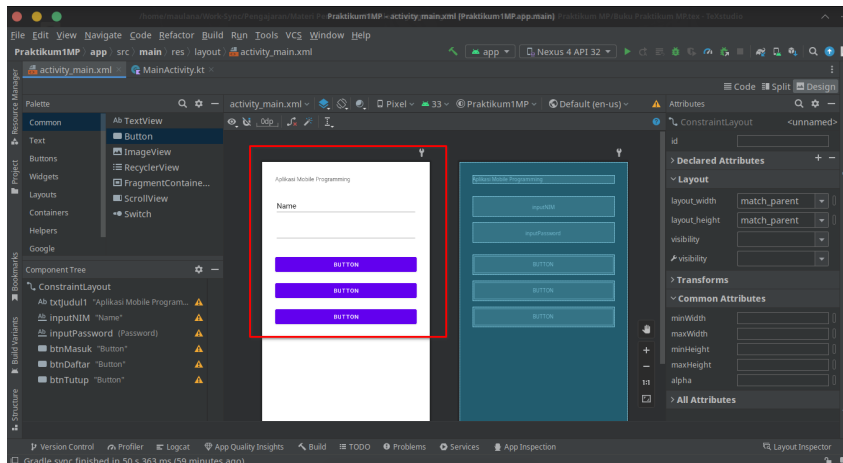
22. Error akan hilang. Namun tampilan terlalu jelek. Untuk mengatur Lebar Objek cukup klik Object → Attribute → layout_width



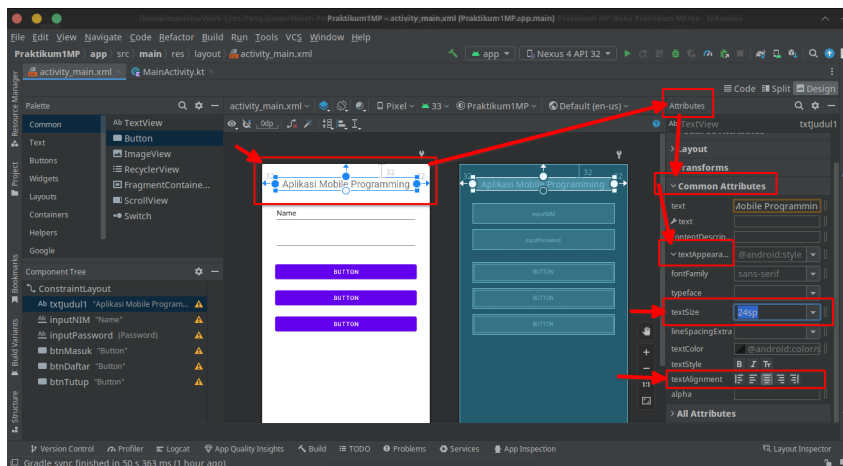
23. Ubah dari wrap_content ke 0dp atau match_constraint untuk melebarkan objek selebar constraint



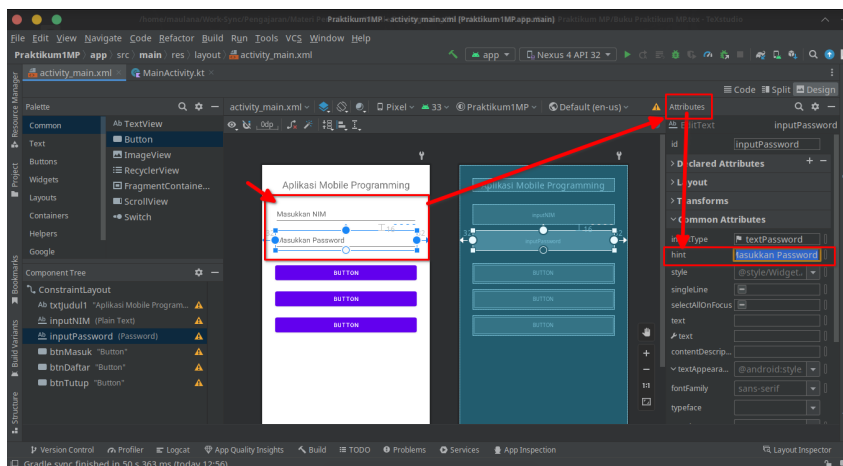
24. Ulangi ke **Semua Objek** hingga terlihat seperti berikut



25. Ubah Letak Teks dari **TextView** Judul menjadi **Center** dan berukuran **24sp**.
Klik Objek → Attribute → **Common Attributes** → **textAppearances**

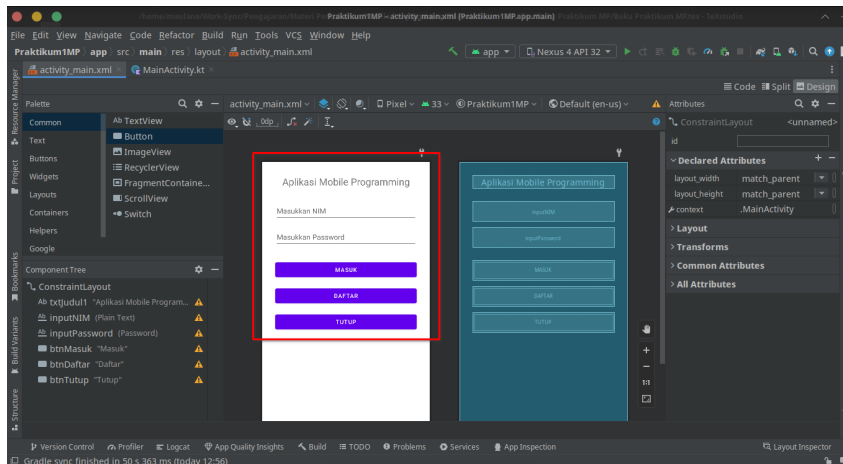


26. Ubah **Hint EditText** (**inputNIM**) dan **EditText** (**inputPassword**) menjadi seperti berikut (Pastikan **Attribute Text** Kosong):



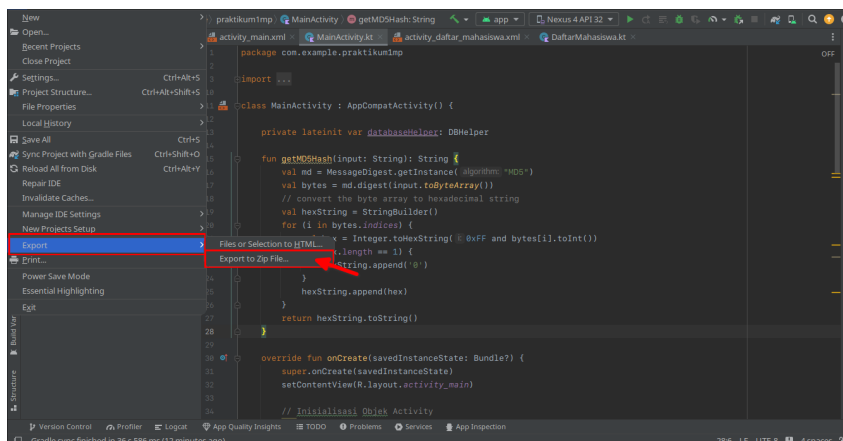
27. Terakhir, ubah **Teks** dari masing-masing **Tombol** menjadi:

- btnMasuk → Masuk
- btnDaftar → Daftar
- btnTutup → Tutup

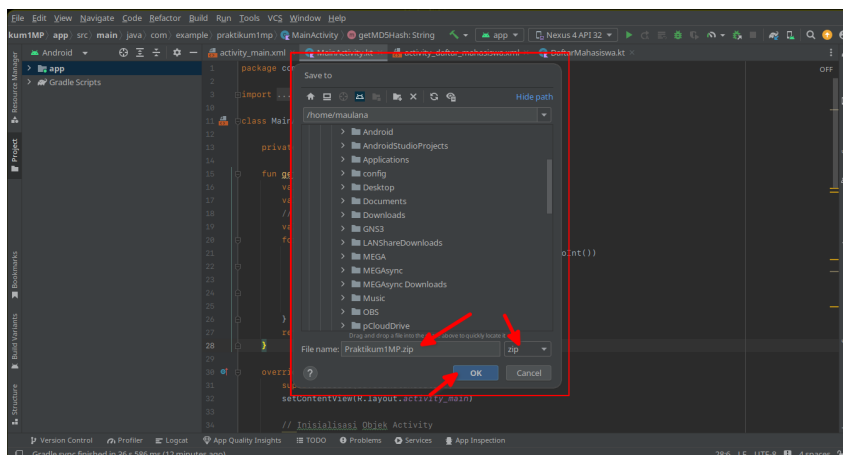


28. Screenshot hasil dan kirimkan ke **E-Learning**

29. Untuk mengirimkannya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



30. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**



Bab 3

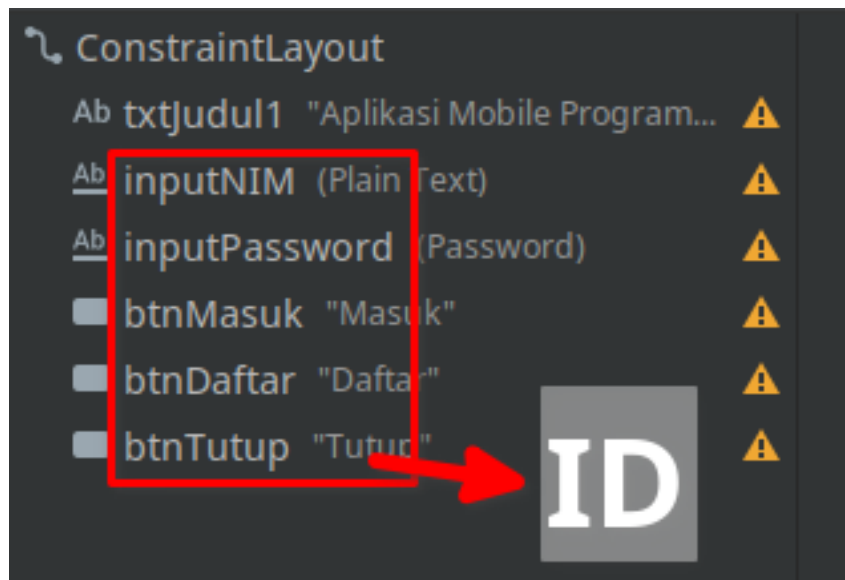
Praktikum 3

3.1 Pemrograman Halaman Login

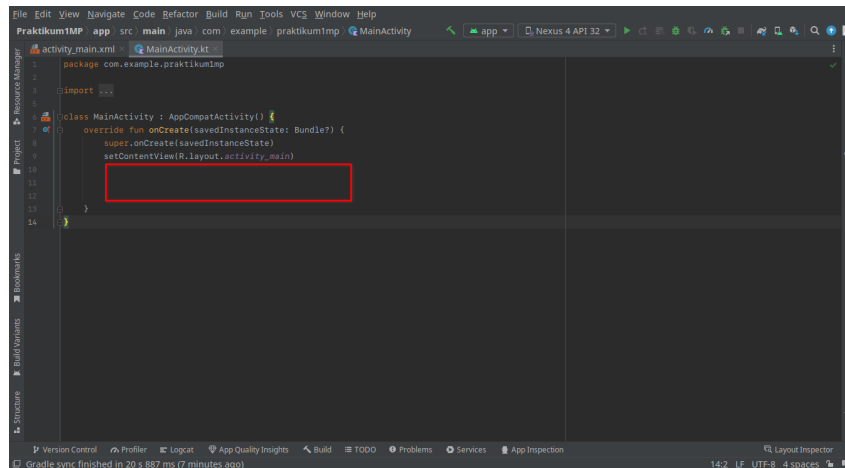
Di bagian ini mahasiswa akan diperkenalkan dengan antarmuka **Android Studio**, sehingga dapat membantu siswa dalam melakukan navigasi yang ada di dalam aplikasi tersebut.

3.2 Tutorial

1. Untuk memulai memrogram objek-objek yang ada di dalam Activity, buka lah projek sebelumnya.
2. Buka `activity_main.xml`, dan pastikan **ID** dari masing-masing objek sudah diatur



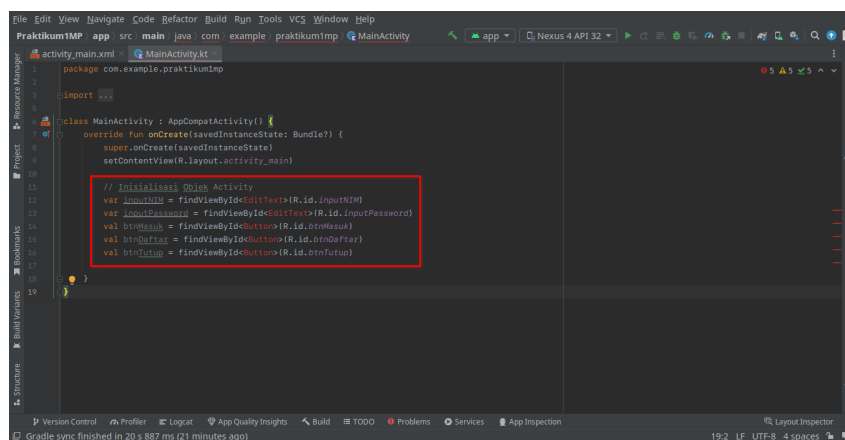
3. Setelah semua objek sudah memiliki ID (Kecuali Judul), berikutnya buka file **Main-Activity.kt** dan tambahkan ruang kosong untuk memasukkan kode fungsional



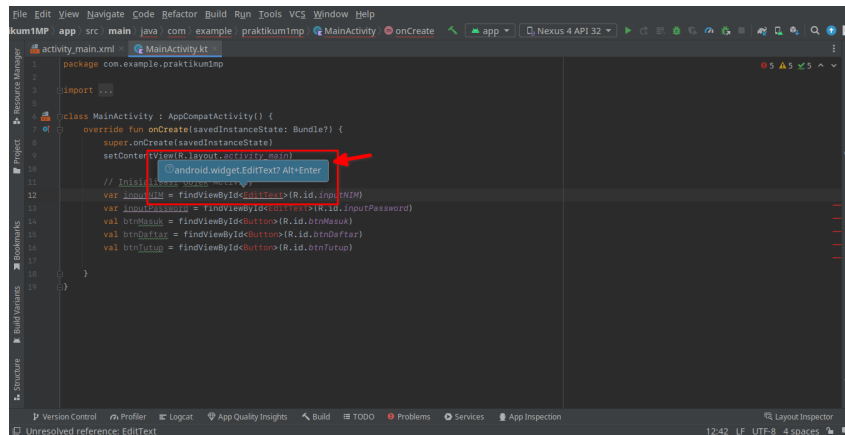
4. Masukkan kode berikut untuk menginisialisasi / mengubah objek menjadi variabel kode

Potongan Kode

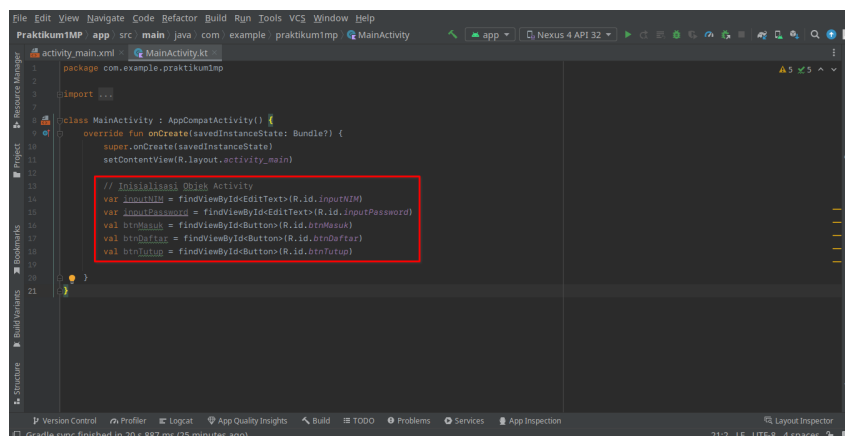
```
// Inisialisasi Objek Activity
var inputNIM = findViewById<EditText>(R.id.inputNIM)
var inputPassword = findViewById<EditText>(R.id.inputPassword)
val btnMasuk = findViewById<Button>(R.id.btnMasuk)
val btnDaftar = findViewById<Button>(R.id.btnDaftar)
val btnTutup = findViewById<Button>(R.id.btnTutup)
```



5. Namun kode ini memiliki **error** cukup **Arahkan** mouse dan **Klik** baris kode yang error, Android Studio akan menawarkan untuk solusi permasalahan. Lalu tekan **Alt+Enter** sesuai arahan rekomendasi itu.



6. Ulangi ke semua kode sehingga tidak muncul **Error**



7. Jika sudah langkah berikutnya adalah membuat kode Aksi untuk **Tombol/But-ton**. Masukkan kode berikut persis di bawah kode **Inisialisasi** sebelumnya. Ruang kosong di dalam Main kode adalah disengaja

Potongan Kode

```

btnMasuk.setOnClickListener {
}
btnDaftar.setOnClickListener {
}
btnTutup.setOnClickListener {
}

```



```

10 override fun onCreate(savedInstanceState: Bundle?) {
11     super.onCreate(savedInstanceState)
12     setContentView(R.layout.activity_main)
13
14     // Inisialisasi Objek Activity
15     var inputNIM = findViewById<EditText>(R.id.inputNIM)
16     var inputPassword = findViewById<EditText>(R.id.inputPassword)
17     val btnMasuk = findViewById<Button>(R.id.btnMasuk)
18     val btnDaftar = findViewById<Button>(R.id.btnDaftar)
19     val btnTutup = findViewById<Button>(R.id.btnTutup)
20
21     btnMasuk.setOnClickListener { @View()
22     }
23     btnDaftar.setOnClickListener { @View()
24     }
25     btnTutup.setOnClickListener { @View()
26     }
27 }

```

8. Berikutnya mengisikan fungsi-fungsi di masing-masing Tombol. Di mulai dari **btnTutup**. Masukkan kode berikut:

```

// Perintahkan Activity untuk Selesai
finish()

```

```

21 btnMasuk.setOnClickListener { @View()
22 }
23 btnDaftar.setOnClickListener { @View()
24 }
25 btnTutup.setOnClickListener { @View()
26     // Perintahkan Activity untuk Selesai
27     finish()
28 }

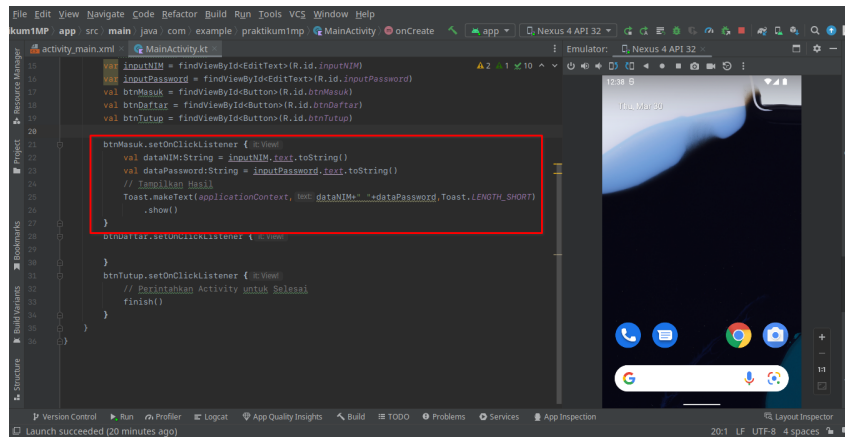
```

9. Kode berikutnya adalah untuk mengambil data login dari **EditText** yang ditempatkan di **activity_main.xml**. Namun saat ini kode hanya bisa menampilkan saja.

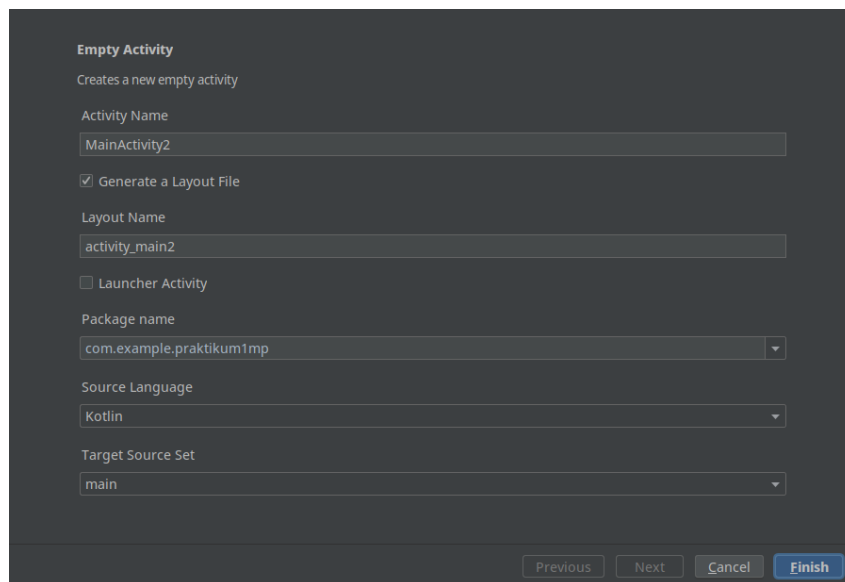
```

val dataNIM:String = inputNIM.text.toString()
val dataPassword:String = inputPassword.text.toString()
// Tampilkan Hasil
Toast.makeText(applicationContext,dataNIM+" "+dataPassword,Toast.LENGTH_SHORT)
.show()

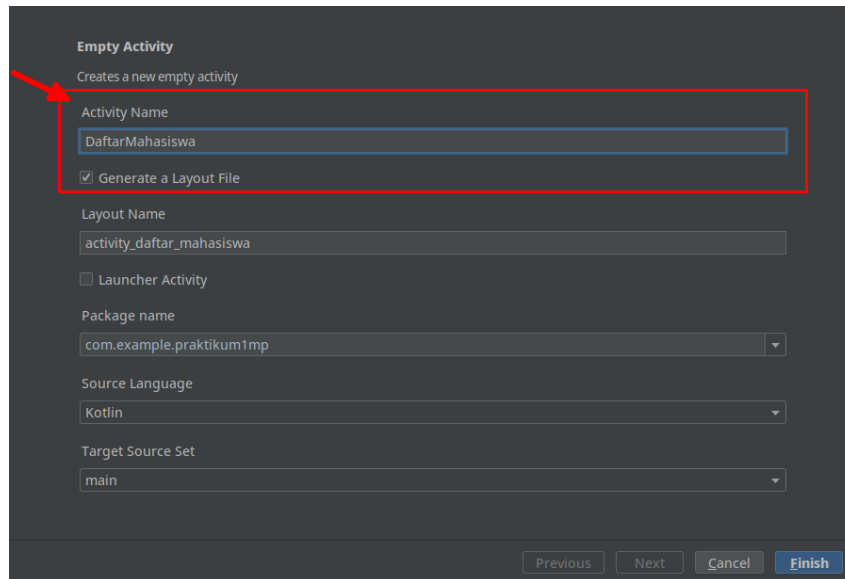
```



10. Sebelum memasukkan kode terakhir, buatlah **Activity** baru dengan cara: Buka Panel **Project** → Klik Kanan Folder **app** → Pilih **new** → Pilih **Activity** → Pilih **Empty Activity**. Maka muncul Window untuk membuat **Activity** baru



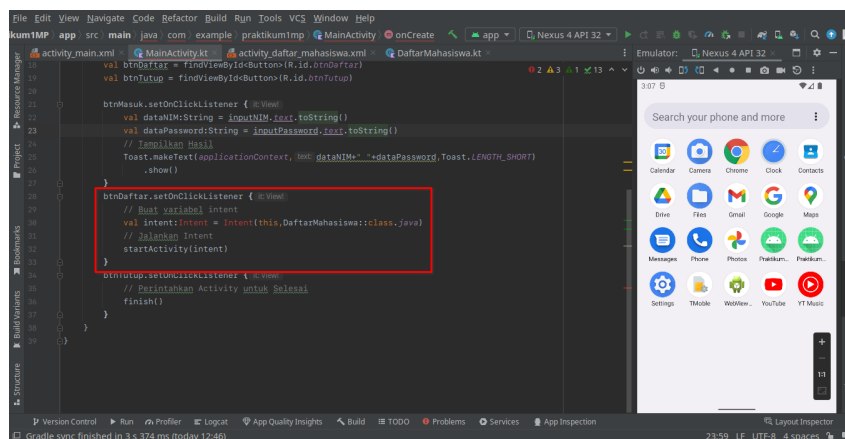
11. Ubah **MainActivity2** menjadi **DaftarMahasiswa** di bagian **Activity Name**. Sisanya akan mengikuti. Lalu klik **Finish**



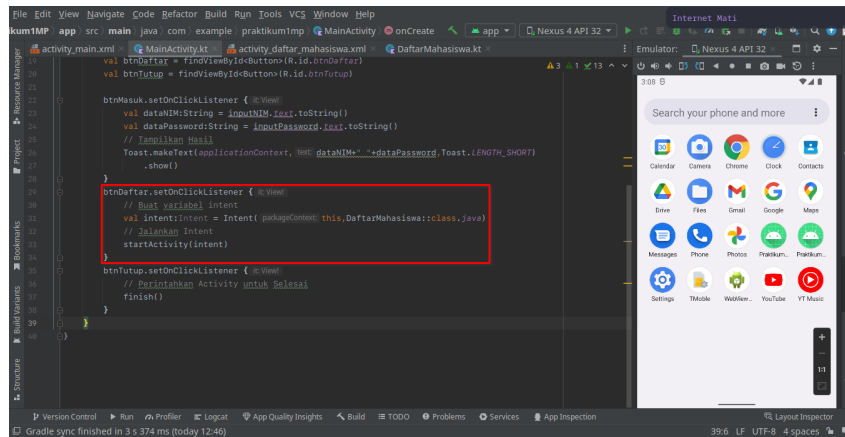
12. Android Studio akan menampilkan layout dan kode kotlin **Activity** baru ini. Kembali buka **MainActivity.kt**

13. Masukkan kode baru untuk tombol **btnDaftar** dengan kode seperti berikut:
Potongan Kode

```
// Buat variabel intent
val intent:Intent = Intent(this,DaftarMahasiswa::class.java)
// Jalankan Intent
startActivity(intent)
```



14. Kode akan mengalami **Error** di bagian Intent, gunakan **Alt+Enter** untuk menyelesaikan masalah **import**



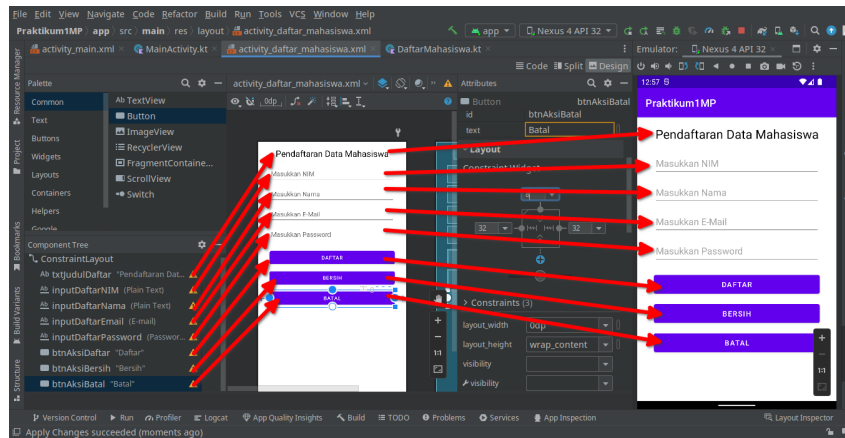
15. Tes aplikasi dengan menjalankannya ke **Emulator**

16. Jika sudah berhasil, tambahkan objek-objek ke layout **activity_daftar_mahasiswa.xml** seperti berikut:

- **TextView**
 - ID : txtJudulDaftar
 - Text : Pendaftaran Data Mahasiswa
- **EditText (Plain)**
 - ID : inputDaftarNIM
 - Hint : Masukkan NIM
- **EditText (Plain)**
 - ID : inputDaftarNama
 - Hint : Masukkan Nama
- **EditText (Email)**
 - ID : inputDaftarEmail
 - Hint : Masukkan E-Mail
- **EditText (Password)**
 - ID : inputDaftarPassword
 - Hint : Masukkan Password
- **Button**
 - ID : btnAksiDaftar
 - Text : Daftar
- **Button**
 - ID : btnAksiBersih
 - Text : Bersih

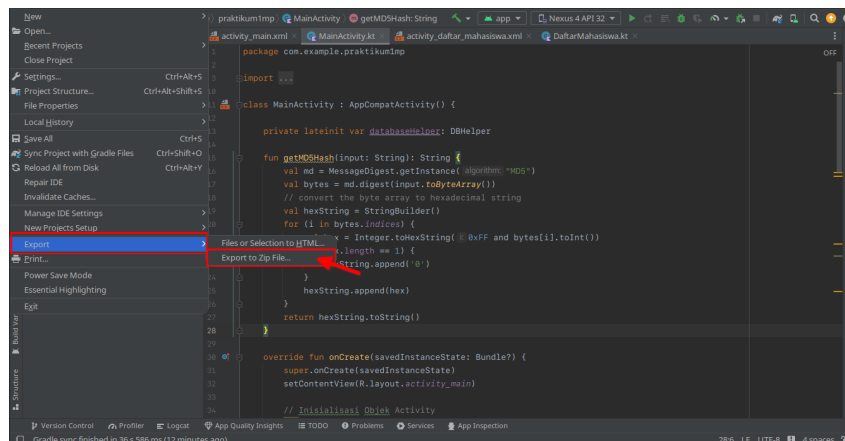
- **Button**

- ID : btnAksiBatal
- Text : Batal

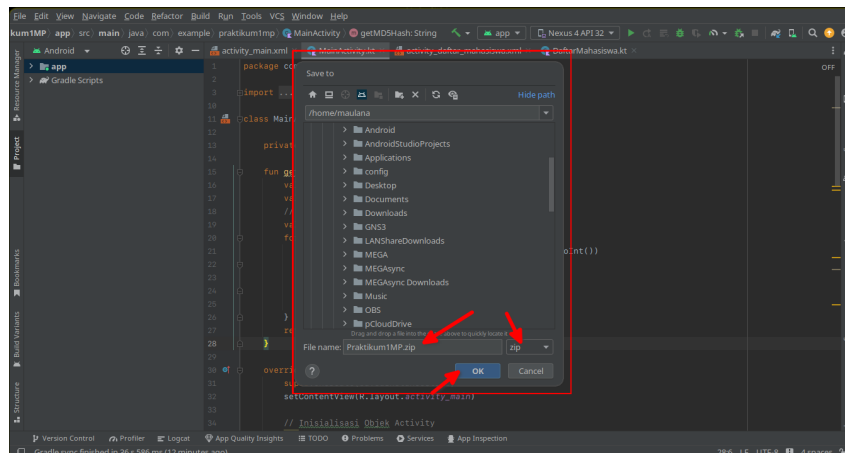


17. Jalankan aplikasi, dan untuk berpindah ke halaman berikutnya klik tombol **Daftar**
 Screenshot hasil dan kirimkan ke **E-Learning**

18. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



19. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**



Bab 4

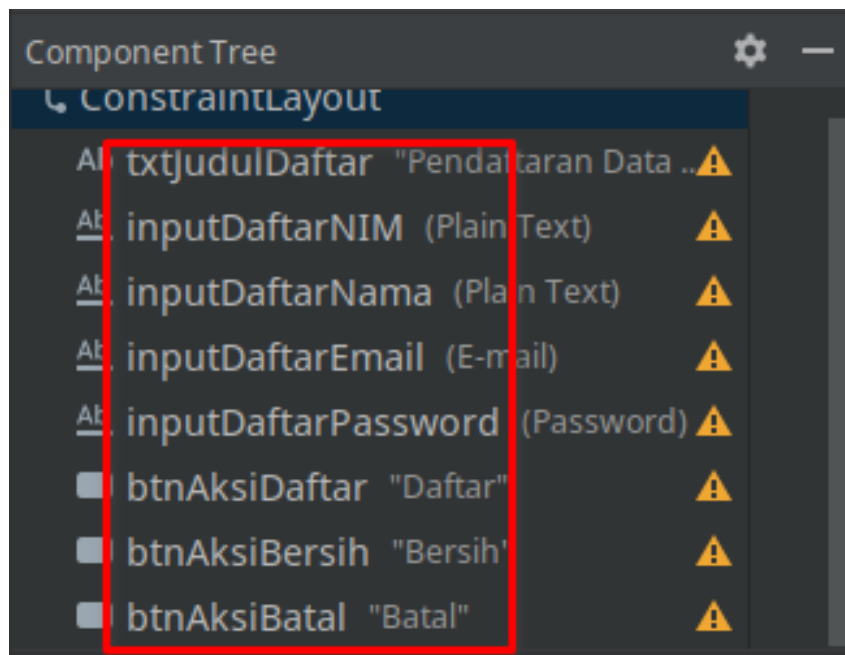
Praktikum 4

4.1 Pemrograman Halaman Pendaftaran

Di bagian ini mahasiswa melanjutkan proyek sebelumnya dengan menambahkan kode aksi di halaman Pendaftaran. Mahasiswa akan diperkenalkan bagaimana melakukan akses data dan menampilkan data.

4.2 Tutorial

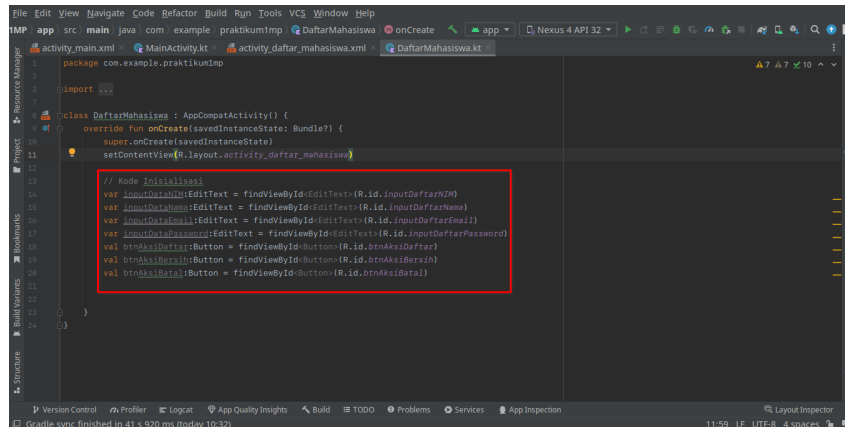
1. Buka proyek sebelumnya yang sudah memiliki **Activity** Pendaftaran
2. Pastikan **ID** yang ada di dalam file `activity_daftar_mahasiswa.xml` persis dengan gambar berikut:



3. Jika **ID** pada layout sudah sesuai, maka buka file `DaftarMahasiswa.kt` dan masukkan kode inisialisasi tepat di bawah baris `setContentview` seperti berikut:

Potongan Kode

```
// Kode Inisialisasi
var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)
```



```
package com.example.praktikumImp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View

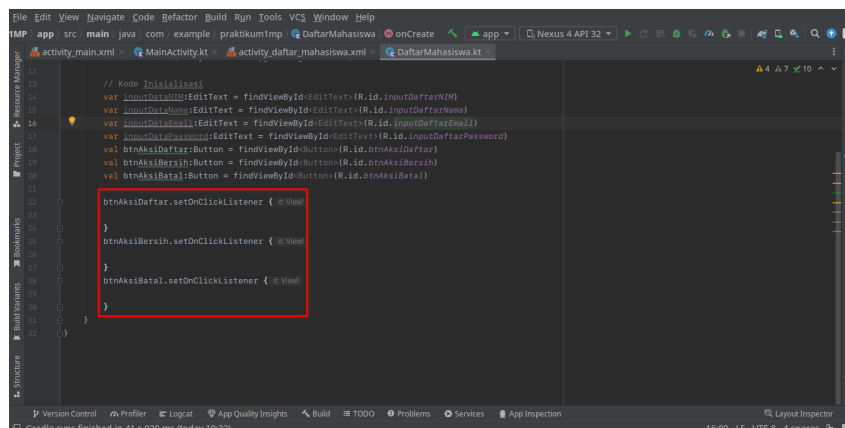
class DaftarMahasiswa : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_daftar_mahasiswa)

        // Kode Inisialisasi
        var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
        var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
        var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
        var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
        val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
        val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
        val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)
    }
}
```

4. Perbaiki **Error** karena **Import** sebelum melanjutkan ke tahap berikutnya
5. Masukkan kode **Event Listener** untuk masing-masing tombol dengan kode seperti berikut:

Potongan Kode

```
btnAksiDaftar.setOnClickListener {
}
btnAksiBersih.setOnClickListener {
}
btnAksiBatal.setOnClickListener {
}
```



```
        // Kode Inisialisasi
        var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
        var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
        var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
        var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
        val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
        val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
        val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)

        btnAksiDaftar.setOnClickListener { @View()
        }
        btnAksiBersih.setOnClickListener { @View()
        }
        btnAksiBatal.setOnClickListener { @View()
        }
    }
}
```

6. Agar tombol-tombol bisa melakukan aksinya, masukkan kode berikut mulai dari tombol **btnAksiBatal**

finish()

Potongan Kode

```

// Kode Inisialisasi
var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)

btnAksiDaftar.setOnClickListener { @View()
}

btnAksiBersih.setOnClickListener { @View()
}

btnAksiBatal.setOnClickListener { @View()
    // Batalan dengan finish
    finish()
}

```

7. Kemudian masukkan kode untuk membersihkan input di **Daftar Mahasiswa**. Kode ini untuk **btnAksiBersih**

inputDataNIM.setText("")
inputDataNama.setText("")
inputDataEmail.setText("")
inputDataPassword.setText("")

Potongan Kode

```

// Kode Inisialisasi
var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)

btnAksiDaftar.setOnClickListener { @View()
}

btnAksiBersih.setOnClickListener { @View()
    inputDataNIM.setText("")
    inputDataNama.setText("")
    inputDataEmail.setText("")
    inputDataPassword.setText("")
}

btnAksiBatal.setOnClickListener { @View()
    // Batalan dengan finish
    finish()
}

```

8. Kode berikutnya digunakan untuk mengambil data dari input, lalu mengamankan password agar tidak menggunakan teks biasa / **plaintext**. Perhatikan kode berikut:

// Ambil Data dari Edit Text
val NIM:String = inputDataNIM.text.toString()
val Nama:String = inputDataNama.text.toString()
val Email:String = inputDataEmail.text.toString()
val Password:String = inputDataPassword.text.toString()

Potongan Kode


```

19 val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
20 val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
21 val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)
22
23 btnAksiDaftar.setOnClickListener { @View()
24     // Ambil Data dari Edit Text
25     val NIM:String = inputDataNIM.text.toString()
26     val Nama:String = inputDataNama.text.toString()
27     val Email:String = inputDataEmail.text.toString()
28     val Password:String = inputDataPassword.text.toString()
29
30 }
31
32 btnAksiBersih.setOnClickListener { @View()
33     // Bersihkan input data
34     inputDataNIM.setText("")
35     inputDataNama.setText("")
36     inputDataEmail.setText("")
37     inputDataPassword.setText("")
38 }
39
40 btnAksiBatal.setOnClickListener { @View()
41     // Batalkan dengan finish
42     finish()
43 }
44 }

```

9. Jika sudah memasukkan kode untuk mengambil data input, maka tahap berikutnya adalah melakukan validasi input tadi dengan kode berikut:

Potongan Kode

```

if(NIM.equals("")||Nama.equals("")||Email.equals("")||Password.equals(""))
{
    Toast.makeText(this,"Input Data Masih Kosong",Toast.LENGTH_SHORT).show()
}
else
{
}

```

```

31
32 btnAksiDaftar.setOnClickListener { @View()
33     // Ambil Data dari Edit Text
34     val NIM:String = inputDataNIM.text.toString()
35     val Nama:String = inputDataNama.text.toString()
36     val Email:String = inputDataEmail.text.toString()
37     val Password:String = inputDataPassword.text.toString()
38
39     if(NIM.equals("")||Nama.equals("")||Email.equals("")||Password.equals(""))
40     {
41         Toast.makeText(this,"Input Data Masih Kosong",Toast.LENGTH_SHORT).show()
42     }
43     else
44     {
45     }
46 }
47
48 btnAksiBersih.setOnClickListener { @View()
49     // Bersihkan input data
50     inputDataNIM.setText("")
51     inputDataNama.setText("")
52     inputDataEmail.setText("")
53 }
54
55 btnAksiBatal.setOnClickListener { @View()
56     // Batalkan dengan finish
57     finish()
58 }
59 }

```

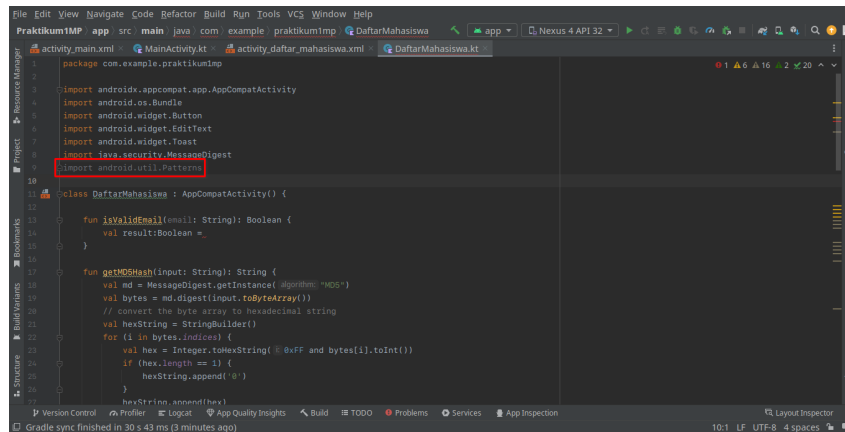
10. Sesudah memberikan logika pengecekan input kosong, berikutnya adalah melakukan validasi E-Mail.
11. Tambahkan **import** untuk pengecekan E-Mail

Potongan Kode

```

import android.util.Patterns

```



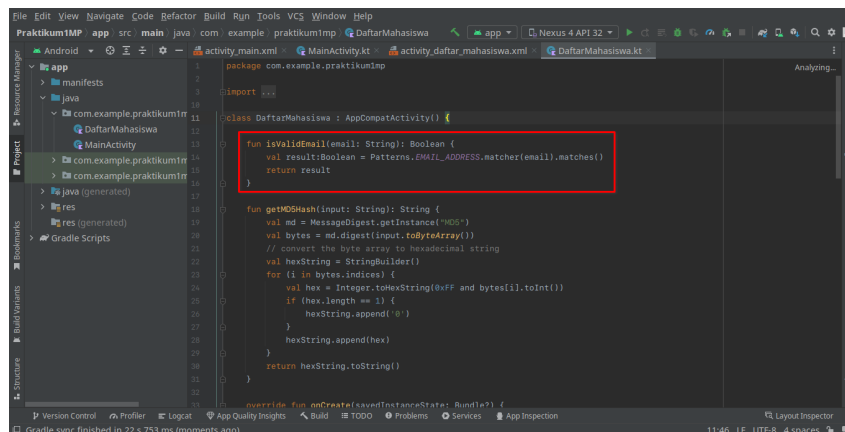
12. Lalu tambahkan kode fungsi pengecekan Email berikut tepat di atas **override fun onCreate** perhatikan gambar untuk membantu

Potongan Kode

```

fun isValidEmail(email:String):Boolean {
    val result:Boolean = Patterns.EMAIL_ADDRESS.matcher(email).matches()
    return result
}

```



13. Kembali ke kode **ELSE** yang dibuat di langkah 9, dan masukkan kode berikut untuk mengecek validitas E-Mail

Potongan Kode

```

if(!isValidEmail(Email))
{
    Toast.makeText(this,"E-Mail Tidak Valid",Toast.LENGTH_SHORT).show()
}
else
{
}

```

```

42     }
43     else
44     {
45         if(!isValidateEmail(email))
46         {
47             Toast.makeText(context, "E-Mail Tidak Valid", Toast.LENGTH_SHORT).show()
48         }
49     }
50 }
51 }
52 }
53 }
54 }
55 }
56 }
57 }
58 }
59 }
60 }
61 }
62 }
63 }
64 }
65 }
66 }
67 }
68 }
69 }
70 }
71 }
72 }
73 }
74 }
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

```

14. Ketika kode untuk verifikasi E-Mail sudah ditambahkan, maka langkah berikutnya adalah melakukan penyamaran **Password** untuk keamanan. Tambahkan kode import berikut di bagian atas

Potongan Kode

```
import java.security.MessageDigest
```

```

1 package com.example.praktikumimp
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.widget.Button
6 import android.widget.EditText
7 import android.widget.Toast
8 import java.security.MessageDigest
9
10 class MainActivity : AppCompatActivity() {
11     fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_daftar_mahasiswa)
14
15         // Kode Inisialisasi
16         val inputNama: EditText = findViewById<EditText>(R.id.inputNama)
17         val inputEmail: EditText = findViewById<EditText>(R.id.inputEmail)
18         val inputPassword: EditText = findViewById<EditText>(R.id.inputPassword)
19     }
20 }

```

15. Setelah itu tambahkan kode untuk melakukan **hashing**. Letakkan di bawah kode pengecekan Email

Potongan Kode

```
fun getMD5Hash(input: String): String {
    val md = MessageDigest.getInstance("MD5")
    val bytes = md.digest(input.toByteArray())
    // Konversi ke Hexa
    val hexString = StringBuilder()
    for (i in bytes.indices) {
        val hex = Integer.toHexString(0xFF and bytes[i].toInt())
        if (hex.length == 1) {
            hexString.append('0')
        }
        hexString.append(hex)
    }
    return hexString.toString()
}
```

```

class DaftarMahasiswa : AppCompatActivity() {
    fun isValidEmail(email: String): Boolean {
        val emailRegex = Regex("^[a-zA-Z0-9-]+@[a-zA-Z0-9-]+\\.([a-zA-Z]{2,6})$")
        return emailRegex.matches(email)
    }

    fun getMD5Hash(input: String): String {
        val md = MessageDigest.getInstance("MD5")
        val bytes = md.digest(input.toByteArray())
        // convert the byte array to hexadecimal string
        val hexString = StringBuilder()
        for (i in bytes.indices) {
            val hex = Integer.toHexString(0xFF and bytes[i].toInt())
            if (hex.length == 1) {
                hexString.append("0")
            }
            hexString.append(hex)
        }
        return hexString.toString()
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
    }
}

```

16. Kode ini kemudian diakses melalui kode yang ditambahkan di dalam **ELSE** dari kode langkah 11. Perhatikan kode dan gambar berikut:

Potongan Kode

```

// Dapatkan Hash Password
val HashedPassword = getMD5Hash>Password)
Toast.makeText(this, "NIM : "+NIM+
    " Nama : "+Nama, Toast.LENGTH_SHORT).show()
Toast.makeText(this, "Email : "+Email+
    " Password : "+HashedPassword, Toast.LENGTH_SHORT).show()

```

```

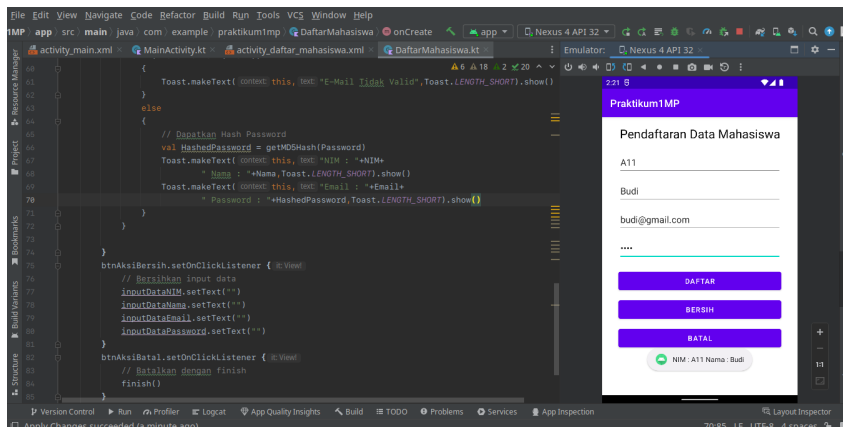
{
    Toast.makeText(this, "Input Data Masih Kosong", Toast.LENGTH_SHORT).show()
} else {
    if (!isValidEmail(Email)) {
        Toast.makeText(this, "E-Mail Tidak Valid", Toast.LENGTH_SHORT).show()
    } else {
        // Dapatkan Hash Password
        val HashedPassword = getMD5Hash>Password)
        Toast.makeText(this, "NIM : "+NIM+
            " Nama : "+Nama, Toast.LENGTH_SHORT).show()
        Toast.makeText(this, "Email : "+Email+
            " Password : "+HashedPassword, Toast.LENGTH_SHORT).show()
    }
}

btnAksiBersih.setOnClickListener { @View
    // Bersihkan input data
    inputDataNIM.setText("")
    inputDataNama.setText("")
    inputDataEmail.setText("")
    inputDataPassword.setText("")
}

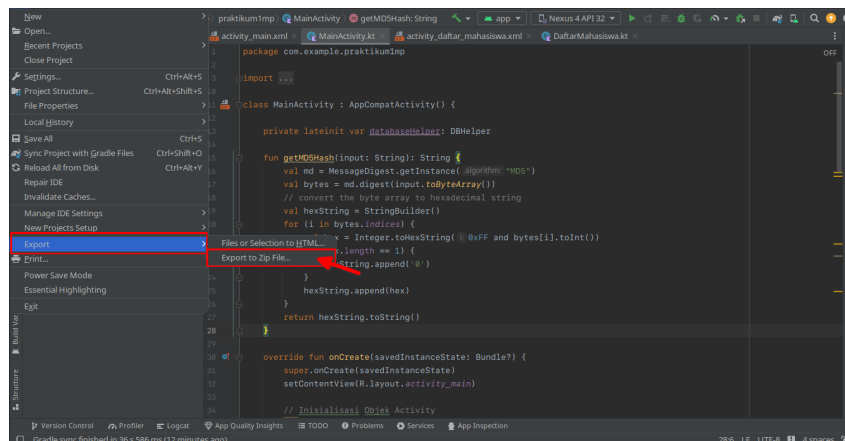
btnAksiBatal.setOnClickListener { @View
    // Batalkan dengan finish
    finish()
}

```

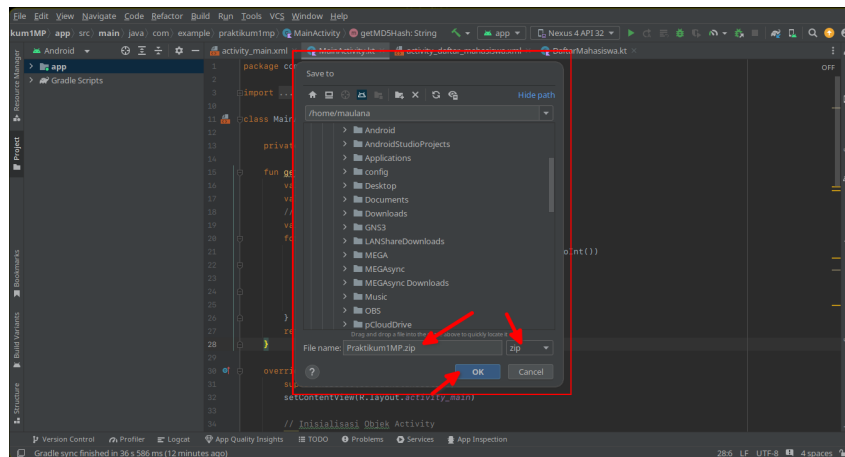
17. Uji coba aplikasi dengan memasukkan data di bagian Pendaftaran



18. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



19. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**



Bab 5

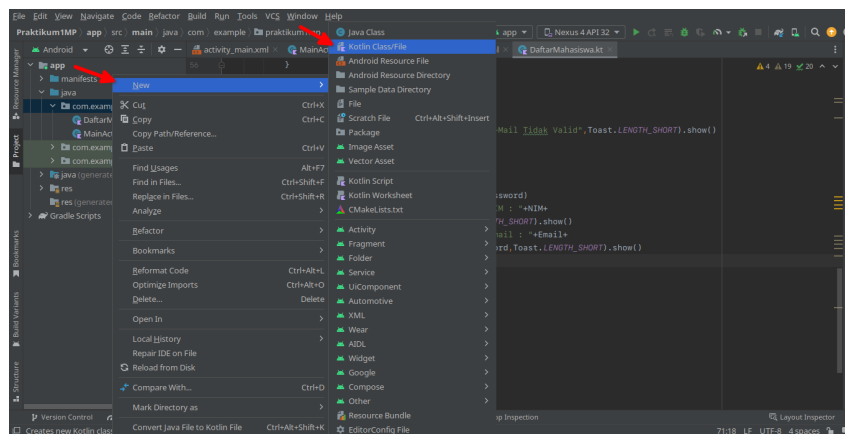
Praktikum 5

5.1 SQLite Insert Data

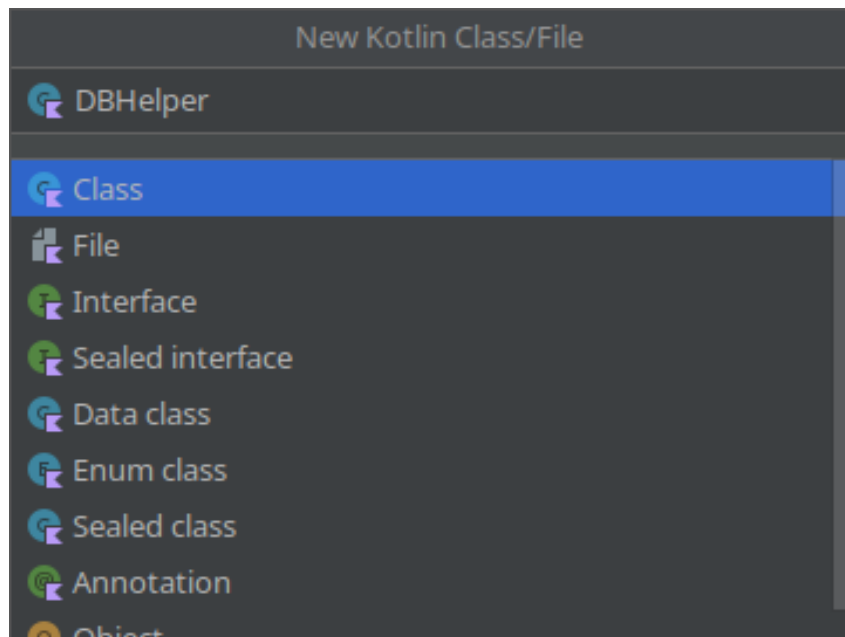
Di bagian ini mahasiswa diajarkan bagaimana melakukan input data melalui Form yang telah dibuat dan mengirimkannya ke database internal SQLite. Mahasiswa diwajibkan menyelesaikan Praktikum 4.

5.2 Tutorial

1. Praktikum dimulai dengan membuka proyek sebelumnya di Android Studio.
2. Sebelum memulai, buatlah satu file **Kotlin** untuk membantu mengatur akses database SQLite. Klik kanan folder **com.example.Praktikum1MP**, pilih **New**, pilih **Kotlin Class/File**



3. Masukkan nama file **DBHelper** lalu tekan **Enter**



4. Masukkan kode berikut untuk file **DBHelper**. Jangan hapus baris kode **package** karena sangat vital.

Potongan Kode

```
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper

class DBHelper(context: Context) : SQLiteOpenHelper(context,
    DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_NAME = "DB_MHS"
        private const val DATABASE_VERSION = 1

        private const val CREATE_TABLE_MY_DATA = "CREATE TABLE TBL_MHS " +
            "(nim TEXT PRIMARY KEY, nama TEXT, email TEXT," +
            "password TEXT)"
    }

    override fun onCreate(db: SQLiteDatabase) {
        db.execSQL(CREATE_TABLE_MY_DATA)
    }

    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
        newVersion: Int) {
        // Untuk upgrade database
    }
}
```

```

1 import android.content.Context
2 import android.database.sqlite.SQLiteDatabase
3 import android.database.sqlite.SQLiteOpenHelper
4
5 class DBHelper(context: Context) : SQLiteOpenHelper(context,
6     DATABASE_NAME, factory=null, DATABASE_VERSION) {
7
8     companion object {
9         private const val DATABASE_NAME = "DB_MHS"
10        private const val DATABASE_VERSION = 1
11    }
12
13    // Definisikan Tabel
14    private const val CREATE_TABLE_MY_DATA = "CREATE TABLE TBL_MHS (" +
15        "(nim TEXT PRIMARY KEY, name TEXT, email TEXT, " +
16        "password TEXT)"
17    )
18
19    override fun onCreate(db: SQLiteDatabase) {
20        db.execSQL(CREATE_TABLE_MY_DATA)
21    }
22
23    override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
24        newVersion: Int) {
25        // Untuk upgrade database
26    }
27
28 }

```

5. Buka file **DaftarMahasiswa.kt**

```

57 else
58 {
59     if (isValidEmail(email))
60     {
61         Toast.makeText(context, R.string.E_Mail_Tidak_Valid, Toast.LENGTH_SHORT).show()
62     }
63     else
64     {
65         // Dapatkan Hash Password
66         val HashedPassword = getMD5Hash>Password)
67         Toast.makeText(context, R.string.HASH : "+NIM+
68             " + Name, Toast.LENGTH_SHORT).show()
69         Toast.makeText(context, R.string.Email : "+Email+
70             " + Password : "+HashedPassword, Toast.LENGTH_SHORT).show()
71     }
72 }
73
74 btnAksiBersih.setOnClickListener { //View
75     // Bersihkan input data
76     inputNama.setText("")
77     inputDaftarMhs.setText("")
78     inputDataEmail.setText("")
79     inputDataPassword.setText("")
80 }
81
82 btnAksiBatal.setOnClickListener { //View
83     // Bersihkan dengan Fungsi

```

6. Tambahkan kode untuk mengakses database, letakkan di bawah kode Class seperti contoh berikut:

Potongan Kode

```
private lateinit var databaseHelper: DBHelper
```

```

1 package com.example.praktikumimp
2 import androidx.appcompat.app.AppCompatActivity
3
4 class DaftarMahasiswa : AppCompatActivity() {
5     private lateinit var databaseHelper: DBHelper
6
7     fun isValidEmail(email: String): Boolean {
8         val result: Boolean = Patterns.EMAIL_ADDRESS.matcher(email).matches()
9         return result
10    }
11
12    fun getMD5Hash(input: String): String {
13        val md = MessageDigest.getInstance("MD5")
14        val bytes = md.digest(input.toByteArray())
15        // convert the byte array to hexadecimal string
16        val hexString = StringBuilder()
17        for (i in bytes.indices) {
18            val hex = Integer.toHexString(0xFF and bytes[i].toInt())
19            if (hex.length == 1) {
20                hexString.append("0")
21            }
22            hexString.append(hex)
23        }
24        return hexString.toString()
25    }
26 }

```

7. Tambahkan lagi inialisasi lokal tepat di bawah kode inialisasi **Widget Layout**. Perhatikan gambar dan kode berikut:

Potongan Kode

```
databaseHelper = DBHelper(this)
```



```

File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Praktikum1MP app src main java com example praktikum1mp DaftarMahasiswa onCreate app Nexus 4 API 32
activity_main.xml MainActivity.kt activity_daftar_mahasiswa.xml DaftarMahasiswa.kt DBHelper.kt
36 override fun onCreate(savedInstanceState: Bundle?) {
37     super.onCreate(savedInstanceState)
38     setContentView(R.layout.activity_daftar_mahasiswa)
39
40     // Kode Inisialisasi
41     val inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
42     val inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
43     val inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
44     val inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)
45     val btnAksiDaftar:Button = findViewById<Button>(R.id.btnAksiDaftar)
46     val btnAksiBersih:Button = findViewById<Button>(R.id.btnAksiBersih)
47     val btnAksiBatal:Button = findViewById<Button>(R.id.btnAksiBatal)
48
49     dbHelper = DBHelper(context)
50
51     btnAksiDaftar.setOnClickListener { @View
52         // Ambil Data dari Edit Text
53         val NIM:String = inputDataNIM.text.toString()
54         val Nama:String = inputDataNama.text.toString()
55         val Email:String = inputDataEmail.text.toString()
56         val Password:String = inputDataPassword.text.toString()
57
58         if(NIM.equals("")||Nama.equals("")||Email.equals("")||Password.equals(""))
59             Toast.makeText(context, "Input Data Masih Kosong", Toast.LENGTH_SHORT).show()
60     }
61 }
Version Control Profiler Logcat App Quality Insights Build TODO Problems Services App Inspection
Gradle sync finished in 33 s 738 ms (today 17:26) 471 LF UTF-8 4 spaces

```

8. Sebelum menambahkan kode untuk kueri, hapus kode berikut dan jangan sentuh variabel **HashedPassword**:

```

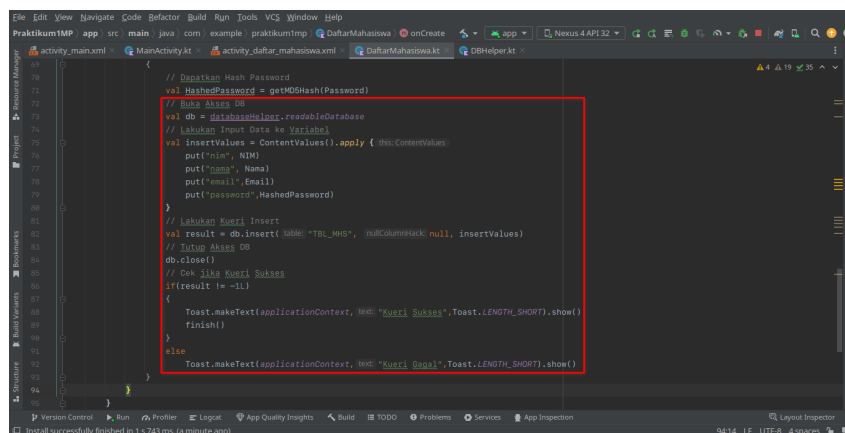
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
Praktikum1MP app src main java com example praktikum1mp DaftarMahasiswa onCreate app Nexus 4 API 32
activity_main.xml MainActivity.kt activity_daftar_mahasiswa.xml DaftarMahasiswa.kt DBHelper.kt
59     Toast.makeText(context, "Input Data Masih Kosong", Toast.LENGTH_SHORT).show()
60 }
61 else
62 {
63     if(!isValidEmail(Email))
64     {
65         Toast.makeText(context, "E-Mail Tidak Valid", Toast.LENGTH_SHORT).show()
66     }
67     else
68     {
69         // Rapatkan Hash Password
70         val HashedPassword = getMD5Hash>Password)
71         Toast.makeText(context, "NIM : "+NIM+
72             "\n Nama : "+Nama, Toast.LENGTH_SHORT).show()
73         Toast.makeText(context, "Email : "+Email+
74             "\n Password : "+HashedPassword, Toast.LENGTH_SHORT).show()
75     }
76 }
77
78 btnAksiBersih.setOnClickListener { @View
79     // Bersihkan input data
80     inputDataNIM.setText("")
81     inputDataNama.setText("")
82     inputDataEmail.setText("")
83     inputDataPassword.setText("")
84 }
Version Control Profiler Logcat App Quality Insights Build TODO Problems Services App Inspection
Gradle sync finished in 33 s 738 ms (today 17:26) 471 LF UTF-8 4 spaces

```

9. Tepat di bawah variabel **HashedPassword**, masukkan kode berikut:

Potongan Kode

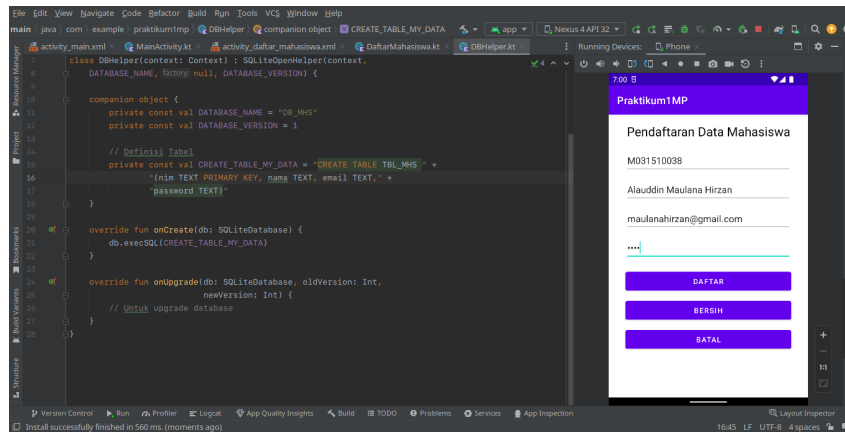
```
// Buka Akses DB
val db = databaseHelper.readableDatabase
// Lakukan Input Data ke Variabel
val insertValues = ContentValues().apply {
    put("nim", NIM)
    put("nama", Nama)
    put("email", Email)
    put("password", HashedPassword)
}
// Lakukan Kueri Insert
val result = db.insert("TBL_MHS", null, insertValues)
// Tutup Akses DB
db.close()
// Cek jika Kueri Sukses
if(result != -1L)
{
    Toast.makeText(applicationContext, "Kueri Sukses", Toast.LENGTH_SHORT).show()
    finish()
}
else
    Toast.makeText(applicationContext, "Kueri Gagal", Toast.LENGTH_SHORT).show()
```



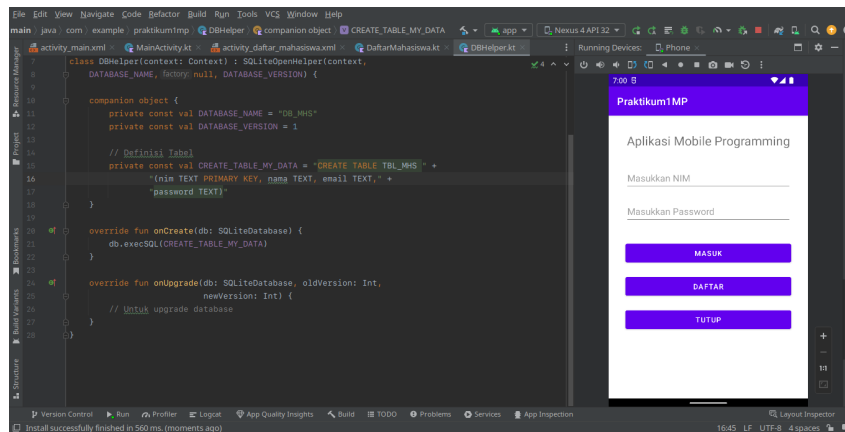
The screenshot shows an IDE window with the following Java code snippet highlighted in a red box:

```
// Dapatkan Hash Password
val HashedPassword = getMD5Hash>Password)
// Buka Akses DB
val db = databaseHelper.readableDatabase
// Lakukan Input Data ke Variabel
val insertValues = ContentValues().apply {
    put("nim", NIM)
    put("nama", Nama)
    put("email", Email)
    put("password", HashedPassword)
}
// Lakukan Kueri Insert
val result = db.insert("TBL_MHS", null, insertValues)
// Tutup Akses DB
db.close()
// Cek jika Kueri Sukses
if(result != -1L)
{
    Toast.makeText(applicationContext, "Kueri Sukses", Toast.LENGTH_SHORT).show()
    finish()
}
else
    Toast.makeText(applicationContext, "Kueri Gagal", Toast.LENGTH_SHORT).show()
```

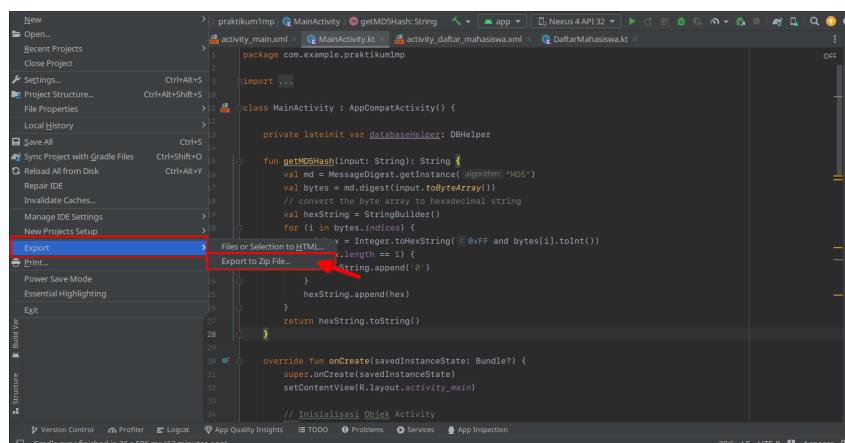
10. Lakukan import untuk **ContentValues** jika terjadi **error**
11. Jalankan **Emulator** terlebih dahulu, lalu buka jalankan aplikasi. Coba kirimkan data melalui form pendaftaran



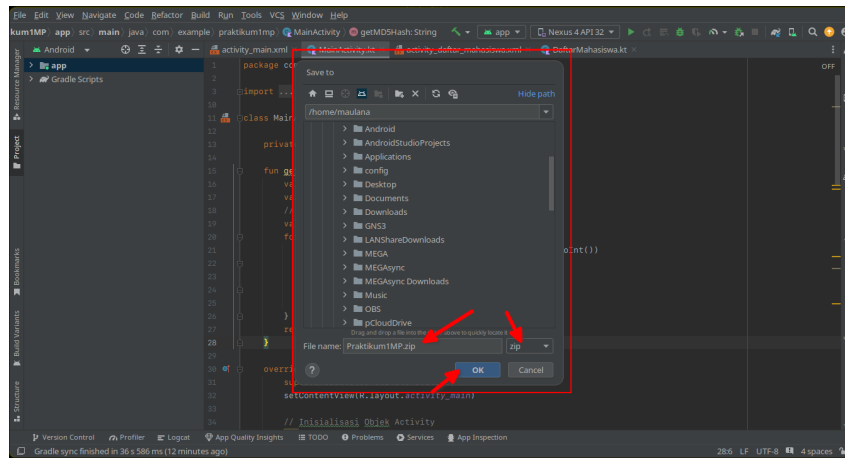
12. Jika kueri berhasil maka akan muncul **Toast** dan halaman akan kembali ke halaman **Login**



13. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



14. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**



Bab 6

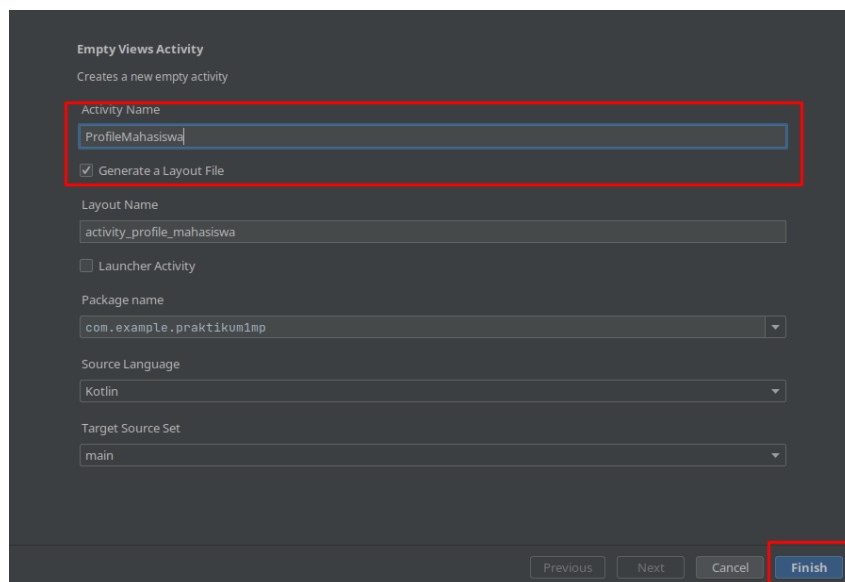
Praktikum 6

6.1 SQLite Read Data

Di bagian ini mahasiswa diajarkan bagaimana melakukan pengambilan data untuk login maupun untuk halaman profil. Mahasiswa diwajibkan untuk menyelesaikan Praktikum 5 sebelum melakukan praktikum ini

6.2 Tutorial

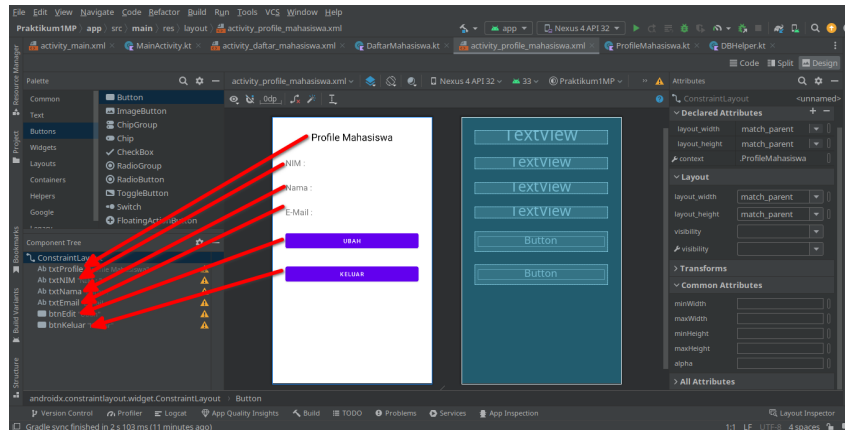
1. Buka kembali Android Studio dan buatlah satu **Activity** baru dengan nama **ProfileMahasiswa**



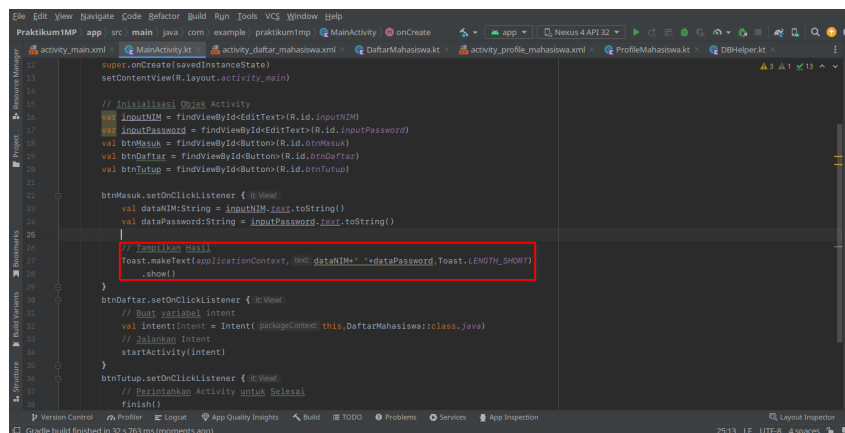
2. Buatlah tampilan halaman untuk **ProfileMahasiswa** di layout **activity_profile_mahasiswa.xml**. Perhatikan daftar objek berikut ini

- TextView : **txtProfile** → Text : **Profile Mahasiswa**
- TextView : **txtNIM** → Text : **NIM :**

- TextView : txtNama → Text : Nama :
- TextView : txtEmail → Text : E-Mail :



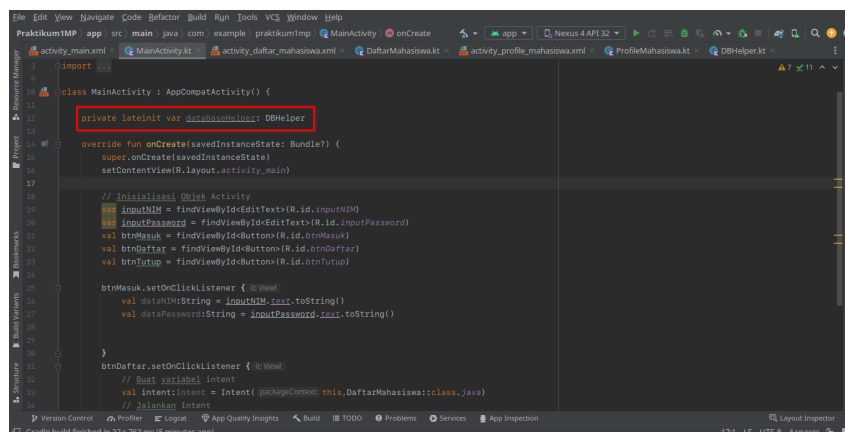
3. Setelah layout untuk **ProfileMahasiswa** sudah jadi, berikutnya buka file **Main-Activity.kt** dan **hapus** bagian kode berikut



4. Setelah dihapus, tambahkan **late init** Database di bawah baris **Class MainActivity**

Potongan Kode

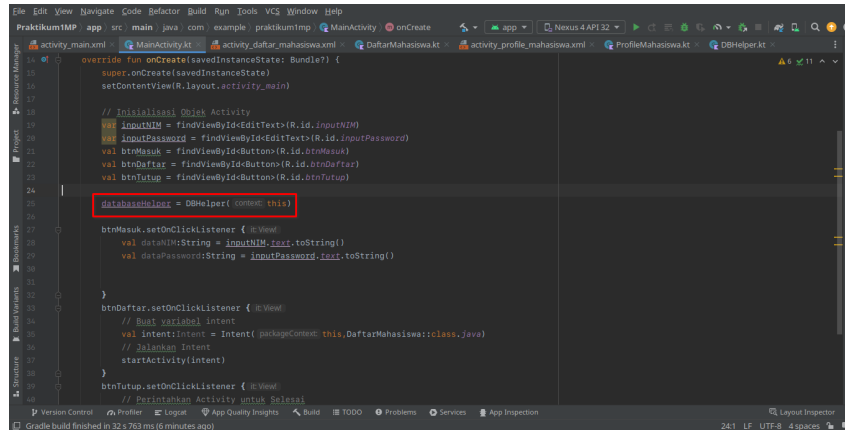
```
private lateinit var databaseHelper: DBHelper
```



5. Inisialisasi database dan letakkan di bawah kode-kode inisialisasi lainnya. Lihat contoh kode dan gambar

Potongan Kode

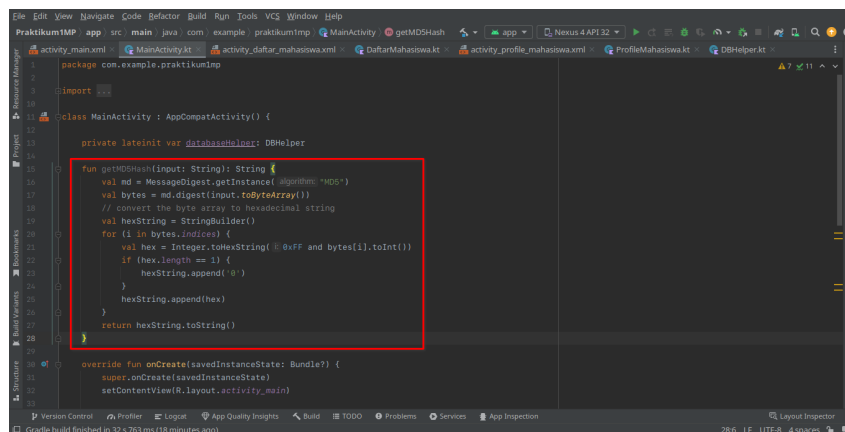
```
databaseHelper = DBHelper(this)
```



6. Setelah itu tambahkan kode untuk melakukan konversi password ke checksum hash. Perhatikan kode berikut dan letakkan di bawah kode **late init**:

Potongan Kode

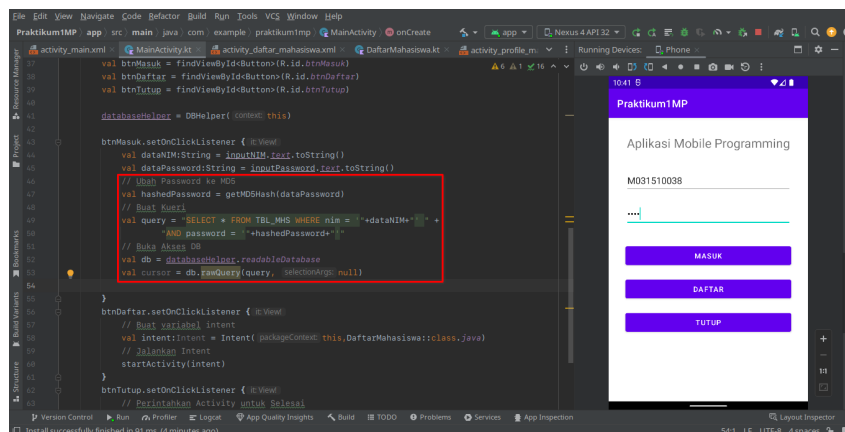
```
fun getMD5Hash(input: String): String {
    val md = MessageDigest.getInstance("MD5")
    val bytes = md.digest(input.toByteArray())
    // Konversi ke Hexa
    val hexString = StringBuilder()
    for (i in bytes.indices) {
        val hex = Integer.toHexString(0xFF and bytes[i].toInt())
        if (hex.length == 1) {
            hexString.append('0')
        }
        hexString.append(hex)
    }
    return hexString.toString()
}
```



7. Berikutnya adalah memasukkan kode untuk verifikasi login, kembali ke bagian yang dihapus tadi (kode `btnMasuk`), dan masukkan kode berikut

Potongan Kode

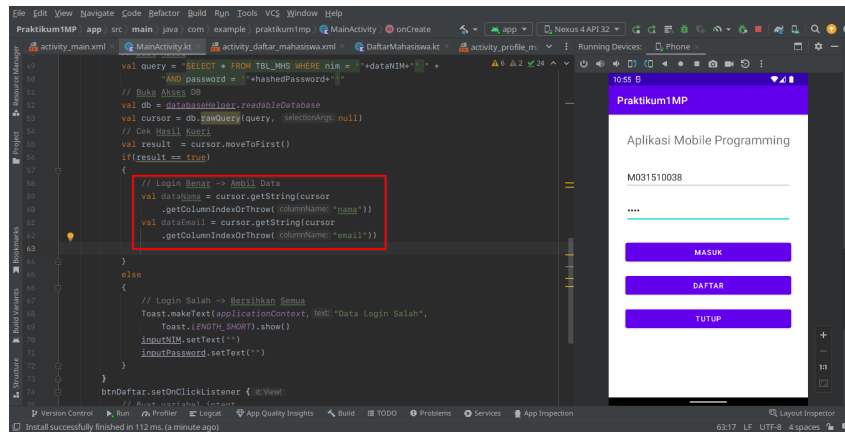
```
// Ubah Password ke MD5
val hashedPassword = getMD5Hash(dataPassword)
// Buat Kueri
val query = "SELECT * FROM TBL_MHS WHERE nim = '"+dataNIM+"' " +
    "AND password = '"+hashedPassword+"'"
// Buka Akses DB
val db = databaseHelper.readableDatabase
val cursor = db.rawQuery(query, null)
```



8. Kemudian lanjutkan kode untuk pengecekan hasil kueri, perhatikan kode dan gambar berikut:

Potongan Kode

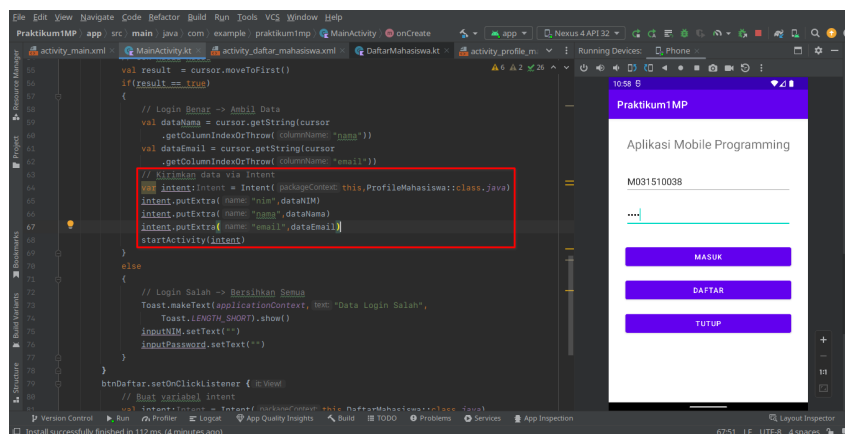
```
// Cek Hasil Kueri
val result = cursor.moveToFirst()
if(result == true)
{
    // Login Benar
}
else
{
    // Login Salah -> Bersihkan Semua
}
{
```

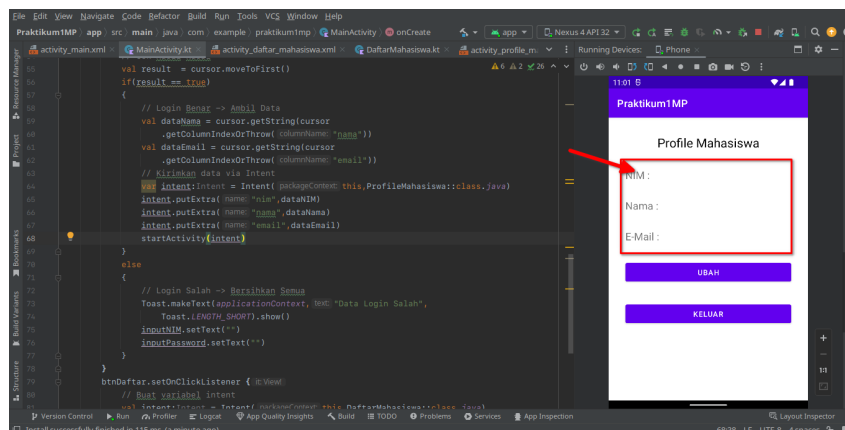
11. Setelah itu masukkan kode berikut untuk mengirimkan data-data tersebut ke halaman ProfileMahasiswa

Potongan Kode

```
// Kirimkan data via Intent
var intent:Intent = Intent(this,ProfileMahasiswa::class.java)
intent.putExtra("nim", dataNIM)
intent.putExtra("nama", dataNama)
intent.putExtra("email", dataEmail)
startActivity(intent)
```



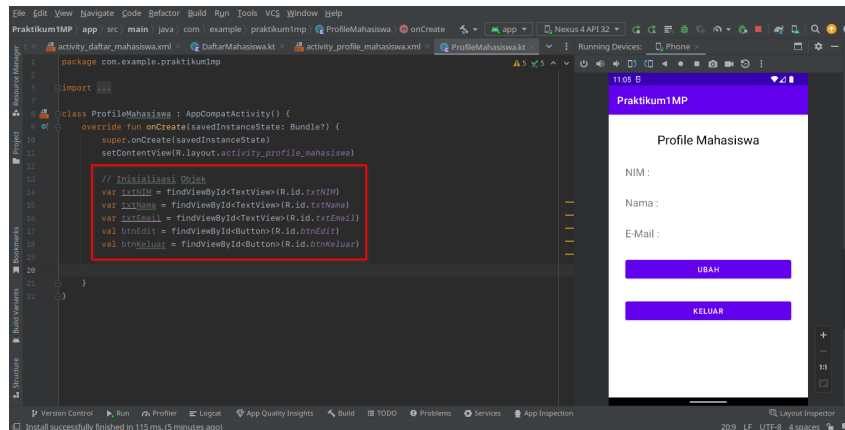
12. Jika dilakukan dengan benar, maka ketika Login aplikasi akan menampilkan Halaman Profile Mahasiswa. Tetapi masih kosong.



13. Agar data dapat ditampilkan dengan benar, masukkan kode **Inisialisasi** terlebih dahulu seperti berikut di file **ProfileMahasiswa.kt**. Perhatikan contoh Kode dan Gambar:

Potongan Kode

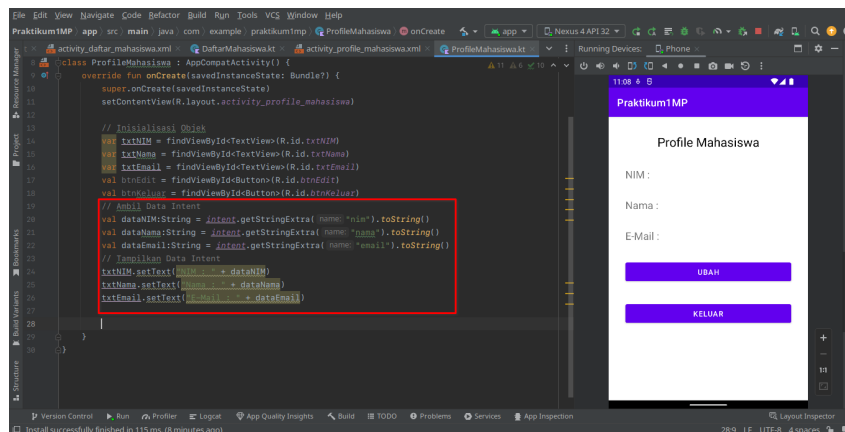
```
// Inisialisasi Objek
var txtNIM = findViewById<TextView>(R.id.txtNIM)
var txtNama = findViewById<TextView>(R.id.txtNama)
var txtEmail = findViewById<TextView>(R.id.txtEmail)
val btnEdit = findViewById<Button>(R.id.btnEdit)
val btnKeluar = findViewById<Button>(R.id.btnKeluar)
```



14. Berikutnya adalah mengambil data **Intent** dan menampilkannya ke layar. Perhatikan kode berikut:

Potongan Kode

```
// Ambil Data Intent
val dataNIM:String = intent.getStringExtra("nim").toString()
val dataNama:String = intent.getStringExtra("nama").toString()
val dataEmail:String = intent.getStringExtra("email").toString()
// Tampilkan Data Intent
txtNIM.setText("NIM : " + dataNIM)
txtNama.setText("Nama : " + dataNama)
txtEmail.setText("E-Mail : " + dataEmail)
```



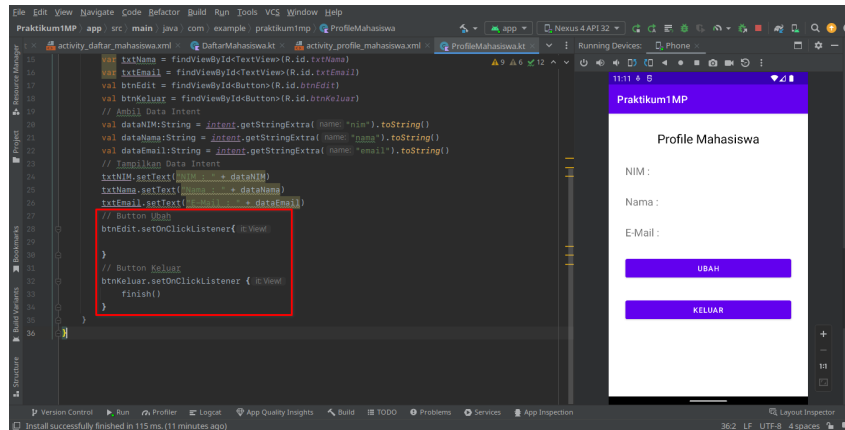
15. Terakhir tambahkan kode **Listener** untuk masing-masing **Button**

Potongan Kode

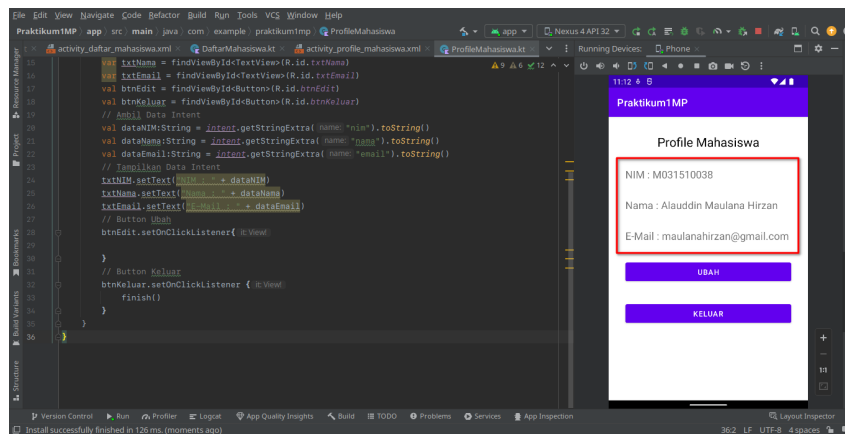
```
// Button Ubah
btnEdit.setOnClickListener{

}

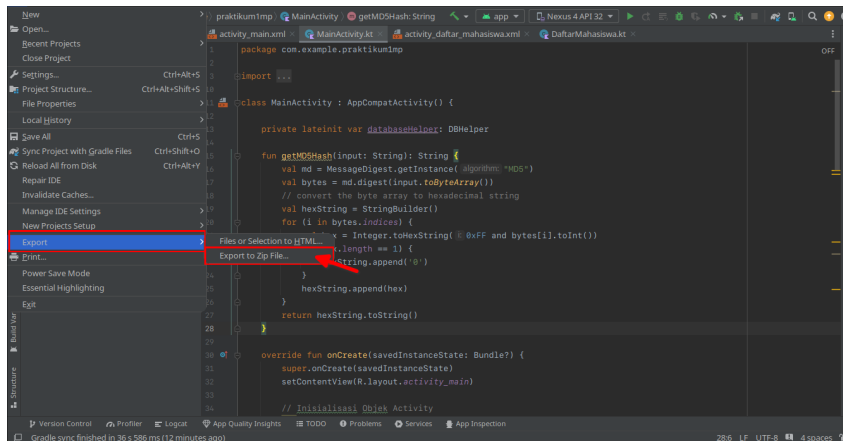
// Button Keluar
btnKeluar.setOnClickListener {
    finish()
}
```



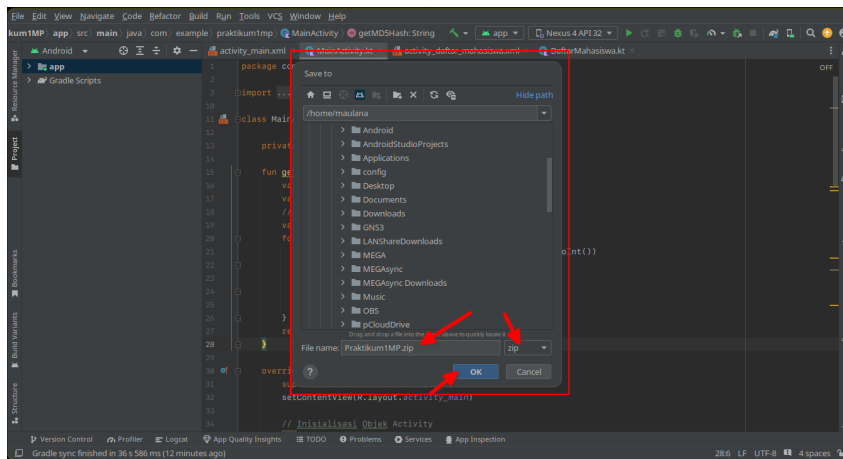
16. Nyalakan Emulator, Jalankan Aplikasi, dan Cobalah Login



17. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



18. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik OK



Bab 7

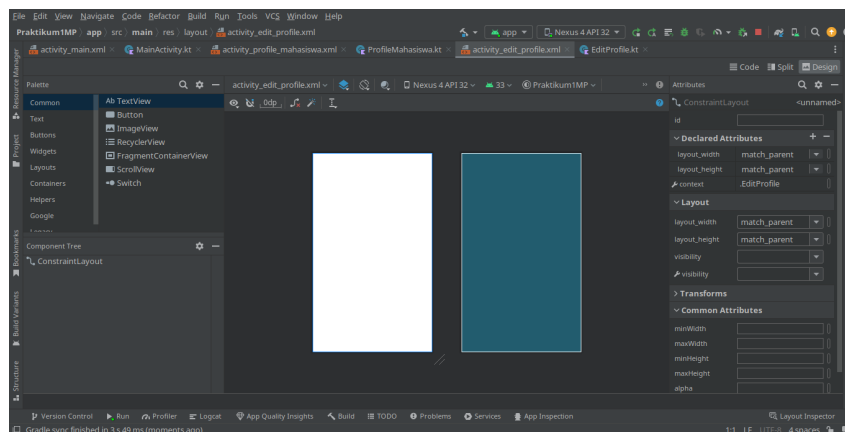
Praktikum 7

7.1 SQLite Update Data

Di bagian ini mahasiswa diajarkan bagaimana melakukan perubahan data yang ada di database. Mahasiswa diwajibkan menyelesaikan Praktikum 6 sebelum melanjutkan praktikum ini

7.2 Tutorial

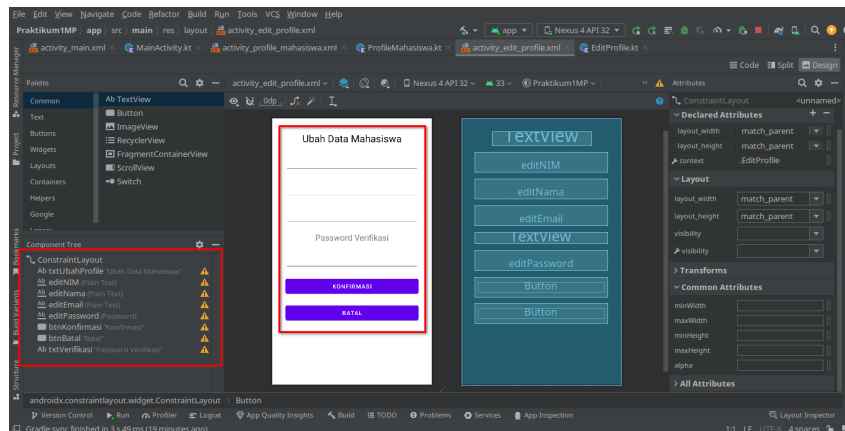
1. Buka Android Studio, dan buatlah satu **Activity Baru** dengan nama **UbahProfile**. Lalu buka layout **activity_edit_profile.xml**



2. Buatlah tampilan seperti berikut:

- TextView → **txtUbahProfile** : Text → **Ubah Data Mahasiswa**
- EditText Plain → **editNIM** : Text → (Kosong)
- EditText Plain → **editNama** : Text → (Kosong)
- EditText Plain → **editEmail** : Text → (Kosong)
- TextView → **txtVerifikasi** : Text → **Password Verifikasi**

- EditText Password → **editPassword** : Text → (Kosong)
- Button → **btnKonfirmasi** : Text → **Konfirmasi**
- Button → **btnBatal** : Text → **Batal**

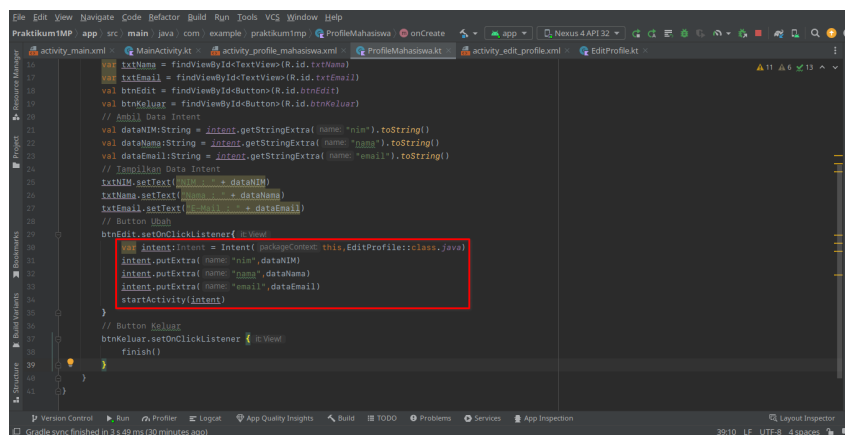


3. Setelah selesai dengan tampilan, kembali ke **ProfileMahasiswa** dan tambahkan kode berikut untuk **btnEdit**

```

Potongan Kode
var intent:Intent = Intent(this,EditProfile::class.java)
intent.putExtra("nim",dataNIM)
intent.putExtra("nama",dataNama)
intent.putExtra("email",dataEmail)
startActivity(intent)

```



4. Berikutnya buka file **EditProfile.kt** kembali, dan tambahkan kode inialisasi seperti contoh berikut:

```

Potongan Kode
// Kode Inialisasi
var editNIM = findViewById<EditText>(R.id.editNIM)
var editNama = findViewById<EditText>(R.id.editNama)
var editEmail = findViewById<EditText>(R.id.editEmail)
var editPassword = findViewById<EditText>(R.id.editPassword)

```

```

1 package com.example.praktikumimp
2
3 import androidx.appcompat.app.AppCompatActivity
4
5 class EditProfile : AppCompatActivity() {
6     override fun onCreate(savedInstanceState: Bundle?) {
7         super.onCreate(savedInstanceState)
8         setContentView(R.layout.activity_edit_profile)
9
10        // Kode Inisialisasi
11        var editNIM = findViewById<EditText>(R.id.editNIM)
12        var editNama = findViewById<EditText>(R.id.editNama)
13        var editEmail = findViewById<EditText>(R.id.editEmail)
14        var editPassword = findViewById<EditText>(R.id.editPassword)
15    }
16 }

```

5. Tambahkan kode berikut untuk mengambil data dari Intent dan mengunci akses editNIM

Potongan Kode

```

// Ambil dan Tampilkan Data Intent
val dataNIM:String = intent.getStringExtra("nim").toString()
val dataNama:String = intent.getStringExtra("nama").toString()
val dataEmail:String = intent.getStringExtra("email").toString()
editNIM.setText(dataNIM)
editNama.setText(dataNama)
editEmail.setText(dataEmail)
// Kunci Akses editNIM
editNIM.isFocusable = false
editNIM.isFocusableInTouchMode = false

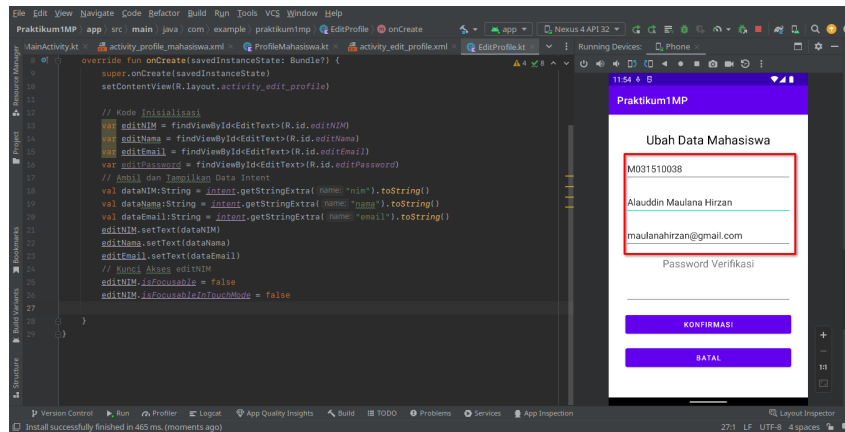
```

```

10        setContentView(R.layout.activity_edit_profile)
11
12        // Kode Inisialisasi
13        var editNIM = findViewById<EditText>(R.id.editNIM)
14        var editNama = findViewById<EditText>(R.id.editNama)
15        var editEmail = findViewById<EditText>(R.id.editEmail)
16        var editPassword = findViewById<EditText>(R.id.editPassword)
17
18        // Ambil dan Tampilkan Data Intent
19        val dataNIM:String = intent.getStringExtra("nim").toString()
20        val dataNama:String = intent.getStringExtra("nama").toString()
21        val dataEmail:String = intent.getStringExtra("email").toString()
22        editNIM.setText(dataNIM)
23        editNama.setText(dataNama)
24        editEmail.setText(dataEmail)
25        // Kunci Akses editNIM
26        editNIM.isFocusable = false
27        editNIM.isFocusableInTouchMode = false
28    }
29 }

```

6. Jalankan Emulator dan Aplikasi untuk mengetes. NIM tidak akan pernah bisa dipencet / klik



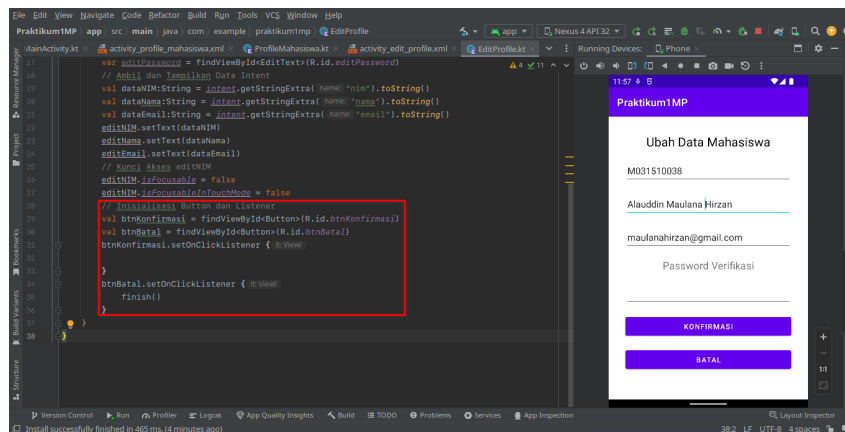
7. Berikutnya tambahkan kode inialisasi Button dan fungsi Listener di masing-masing tombol. Perhatikan kode berikut

```

Potongan Kode
// Inisialisasi Button dan Listener
val btnKonfirmasi = findViewById<Button>(R.id.btnKonfirmasi)
val btnBatal = findViewById<Button>(R.id.btnBatal)
btnKonfirmasi.setOnClickListener {

}
btnBatal.setOnClickListener {
    finish()
}

```

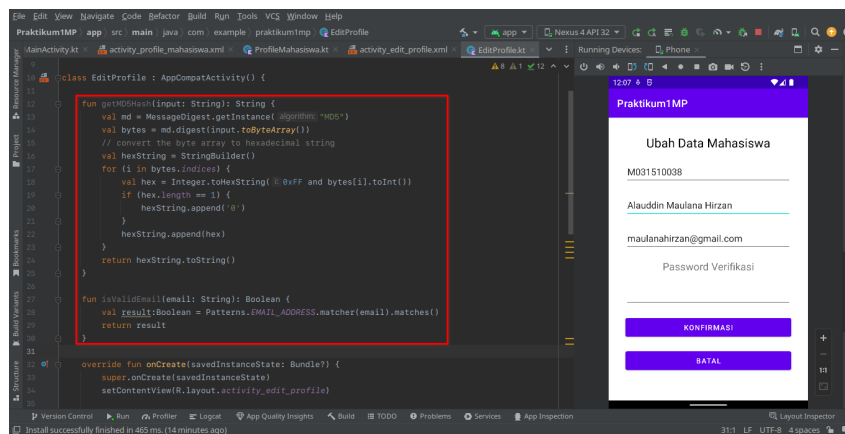


8. Berikutnya masukkan kode untuk mendukung perubahan data seperti **Hashing MD5** dan **E-Mail Checker**. Letakkan dibawah baris kode **Class**. Perhatikan contoh berikut:

Potongan Kode

```
fun getMD5Hash(input: String): String {
    val md = MessageDigest.getInstance("MD5")
    val bytes = md.digest(input.toByteArray())
    // convert the byte array to hexadecimal string
    val hexString = StringBuilder()
    for (i in bytes.indices) {
        val hex = Integer.toHexString(0xFF and bytes[i].toInt())
        if (hex.length == 1) {
            hexString.append('0')
        }
        hexString.append(hex)
    }
    return hexString.toString()
}

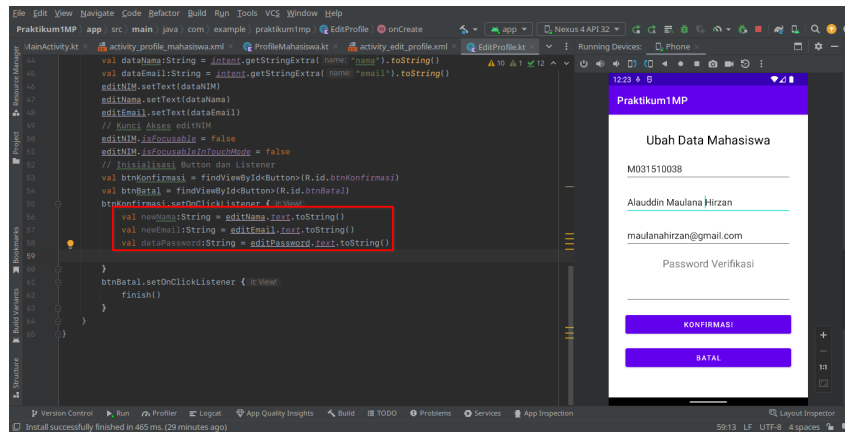
fun isValidEmail(email: String): Boolean {
    val result:Boolean = Patterns.EMAIL_ADDRESS.matcher(email).matches()
    return result
}
```



9. Pindah ke bagian **btnKonfirmasi** dan tambahkan kode berikut untuk mengambil data baru. Perhatikan contoh kode dan gambar:

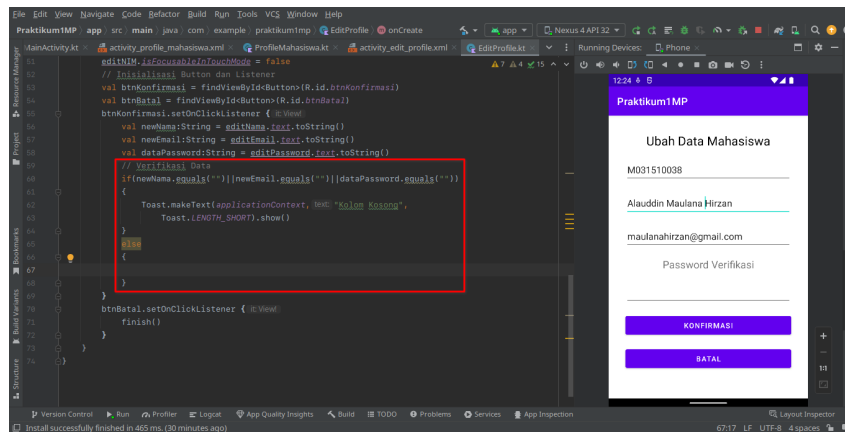
Potongan Kode

```
val newNama:String = editNama.text.toString()
val newEmail:String = editEmail.text.toString()
val dataPassword:String = editPassword.text.toString()
```



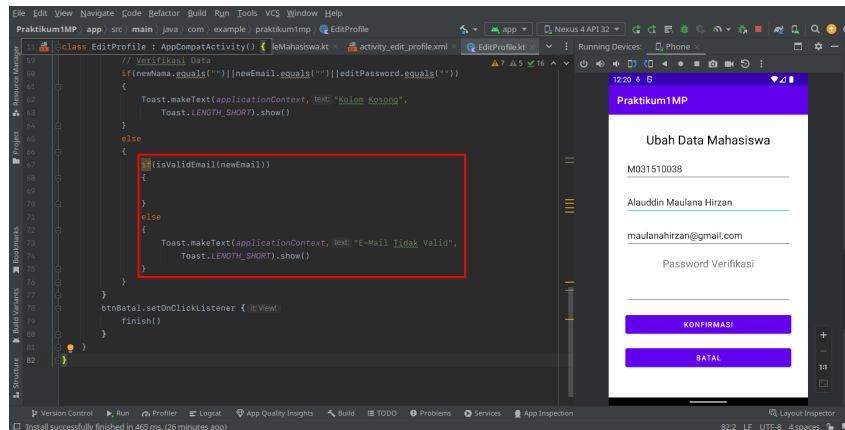
10. Lalu tambahkan kode untuk verifikasi data baru. Perhatikan contoh: **Potongan Kode**

```
// Verifikasi Data
if(newNama.equals("")||newEmail.equals("")||dataPassword.equals(""))
{
    Toast.makeText(applicationContext,"Kolom Kosong",
        Toast.LENGTH_SHORT).show()
}
else
{
}
```



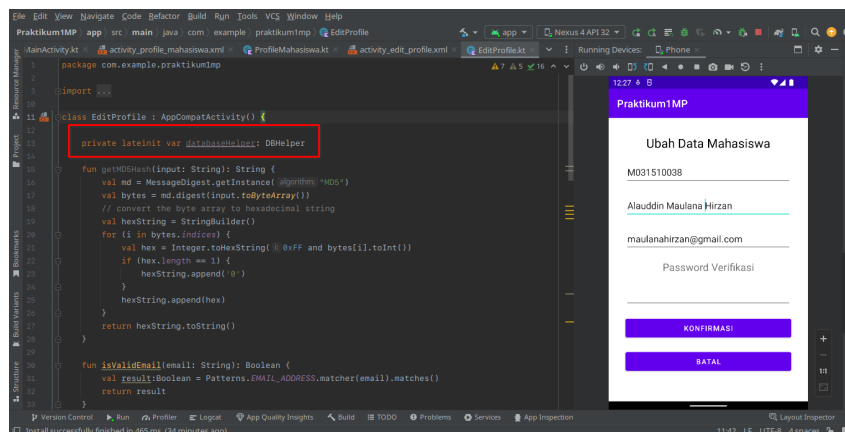
11. Kemudian tambahkan kode verifikasi E-Mail tepat di dalam **ELSE** **Potongan Kode**

```
if(isValidEmail(newEmail))
{
}
else
{
    Toast.makeText(applicationContext,"E-Mail Tidak Valid",
        Toast.LENGTH_SHORT).show()
}
```



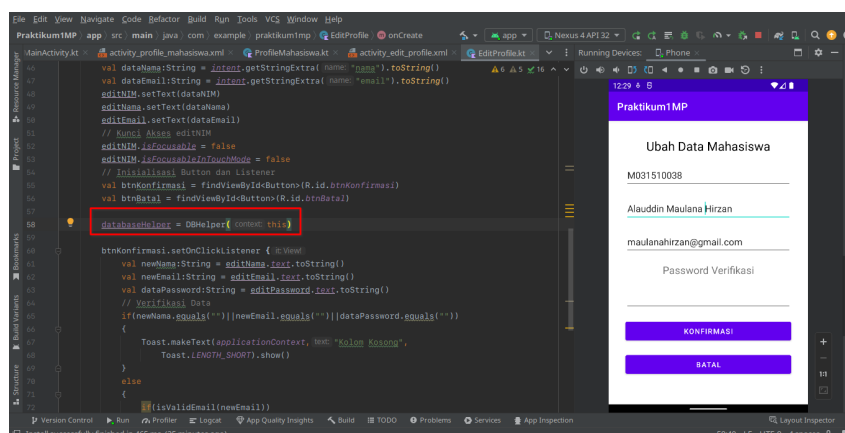
12. Kode hampir siap. Namun sebelum itu, tambahkan kode **late init** lagi tepat di bawah baris kode **Class**

Potongan Kode
`private lateinit var databaseHelper: DBHelper`



13. Kemudian tambahkan kode Inisialisasi DB tepat di bawah kode inisialisasi **Button**. Perhatikan gambar

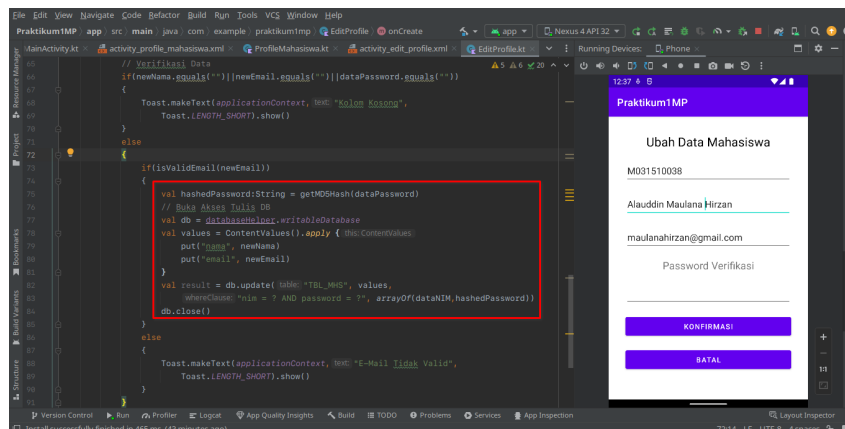
Potongan Kode
`databaseHelper = DBHelper(this)`



14. Kini sudah siap berikutnya menambahkan kode inti **IF** di tombol **btnKonfirmasi** sebelumnya

Potongan Kode

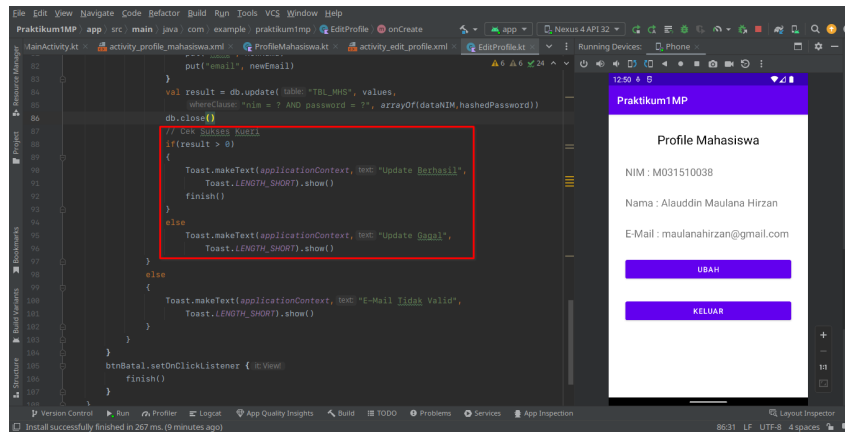
```
val hashedPassword:String = getMD5Hash(dataPassword)
// Buka Akses Tulis DB
val db = databaseHelper.writableDatabase
val values = ContentValues().apply {
    put("nama", newNama)
    put("email", newEmail)
}
val result = db.update("TBL_MHS", values,
    "nim = ? AND password = ?", arrayOf(dataNIM,hashedPassword))
db.close()
```



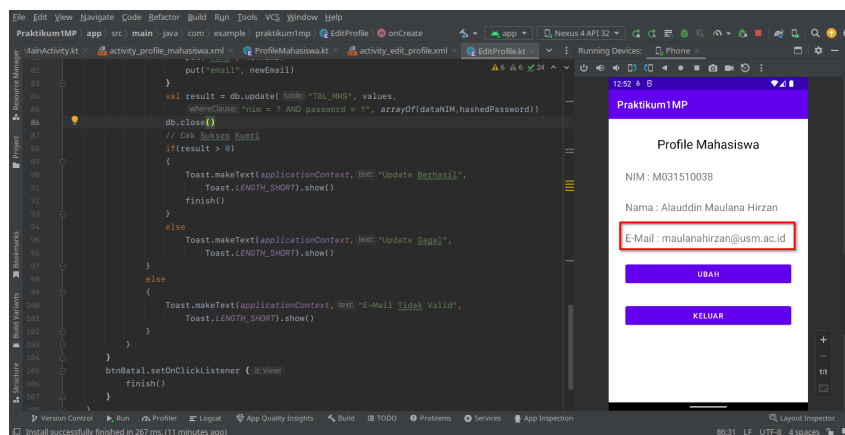
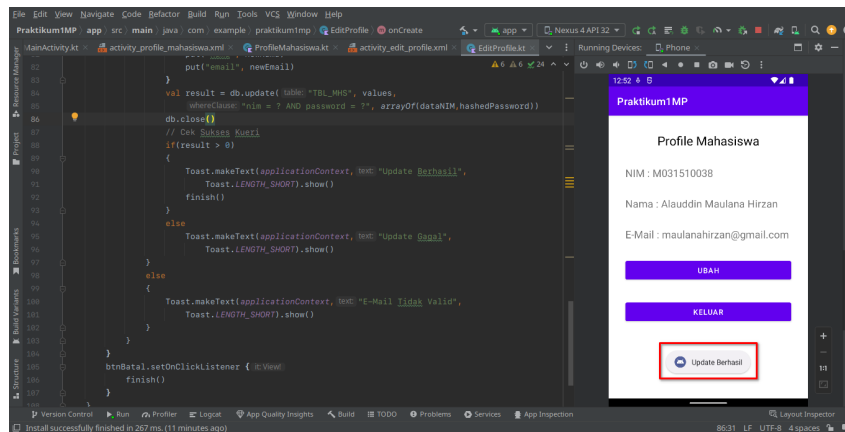
15. Terakhir tambahkan kode pengecekan update tepat di bawah kode sebelumnya. Perhatikan contoh:

Potongan Kode

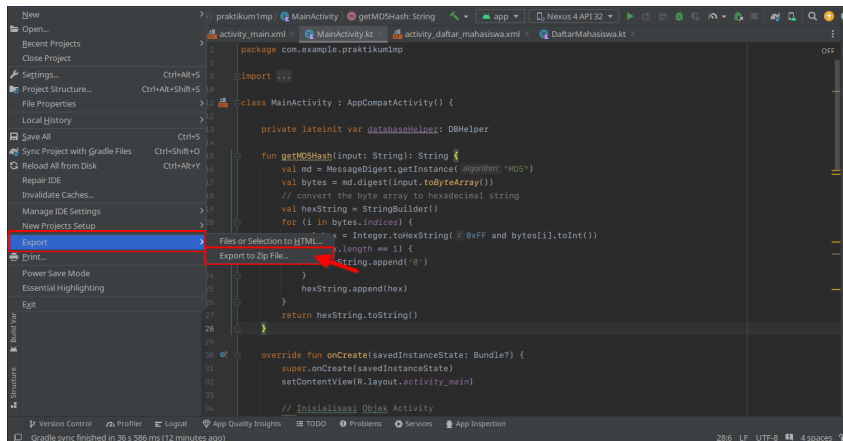
```
// Cek Sukses Kueri
if(result > 0)
{
    Toast.makeText(applicationContext,"Update Berhasil",
        Toast.LENGTH_SHORT).show()
    finish()
}
else
    Toast.makeText(applicationContext,"Update Gagal",
        Toast.LENGTH_SHORT).show()
```



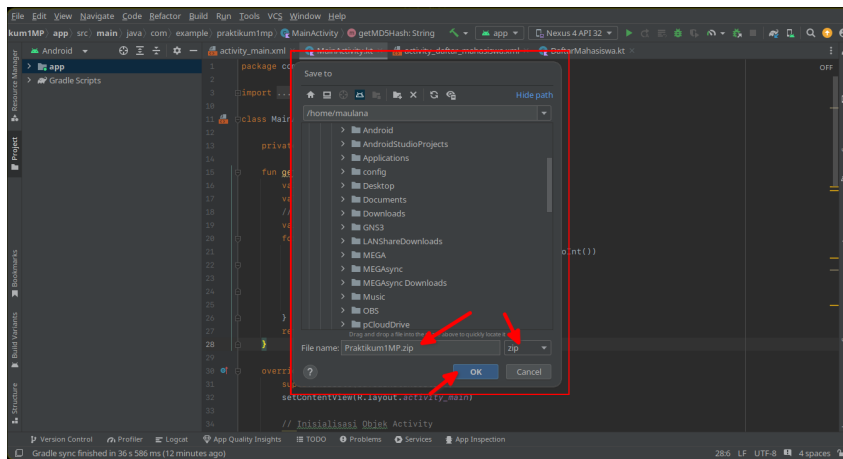
16. Jalankan Emulator dan Aplikasi. Dan Ubah Nama / E-Mail. Pastikan untuk **LOGOUT** setelah berhasil mengubah Data.



17. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



18. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik OK



Bab 8

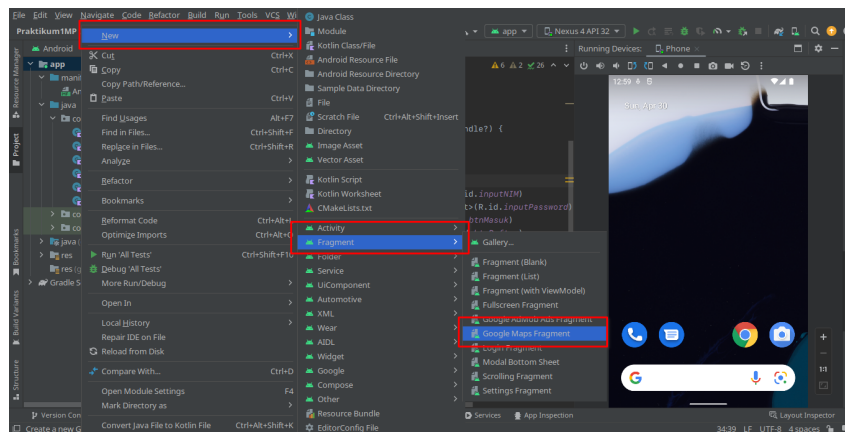
Praktikum 8

8.1 Google Map

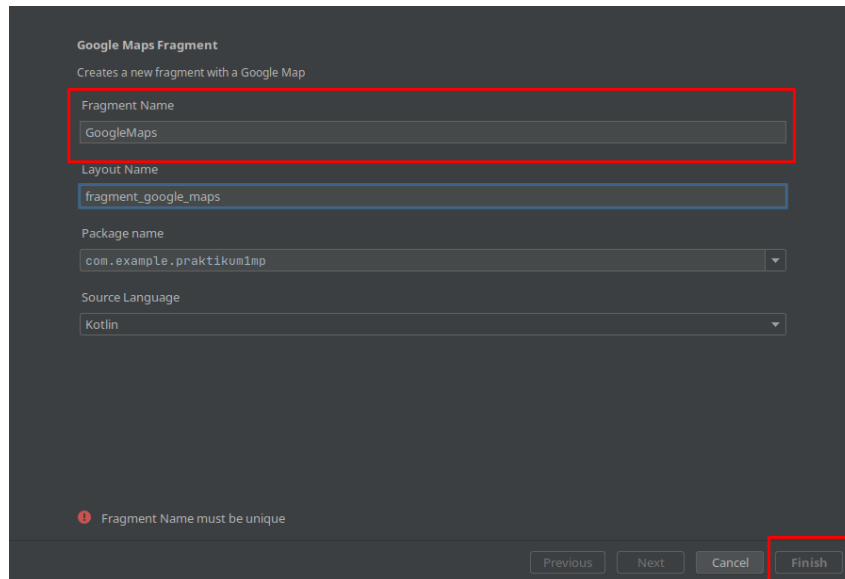
Di bagian ini mahasiswa diajarkan bagaimana cara menambahkan fungsi **Google Maps** di bagian **Profile Mahasiswa**. Mahasiswa diwajibkan untuk menyelesaikan Praktikum 7

8.2 Tutorial

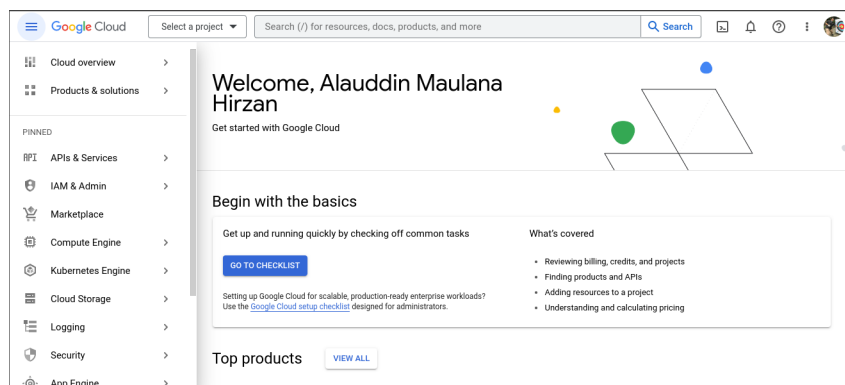
1. Buka Android Studio dan buatlah **Fragment** bukan **Activity** dengan melakukan klik Kanan **app** Project → pilih **New** → pilih **Fragment** dan Klik **Google Maps Fragment**



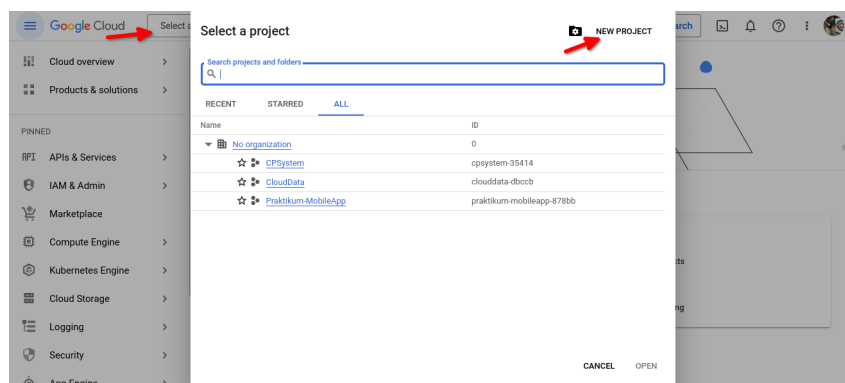
2. Beri nama **GoogleMaps**, dan klik **Finish**



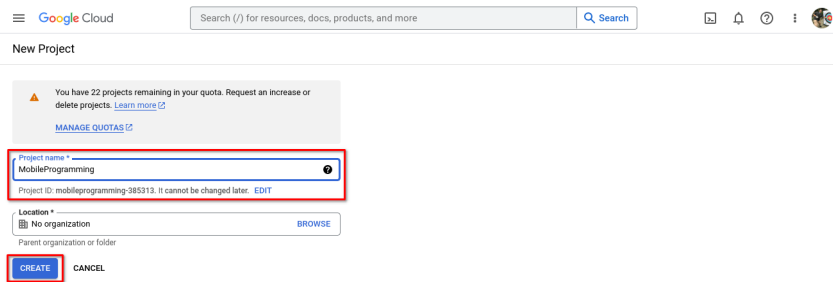
3. Tunggu beberapa saat hingga **Android Studio** selesai melakukan loading.
4. Setelah itu buka website <https://console.cloud.google.com>. Login dengan akun Google pribadi. Jika sudah pastikan berada di halaman Dasbornya seperti gambar berikut:



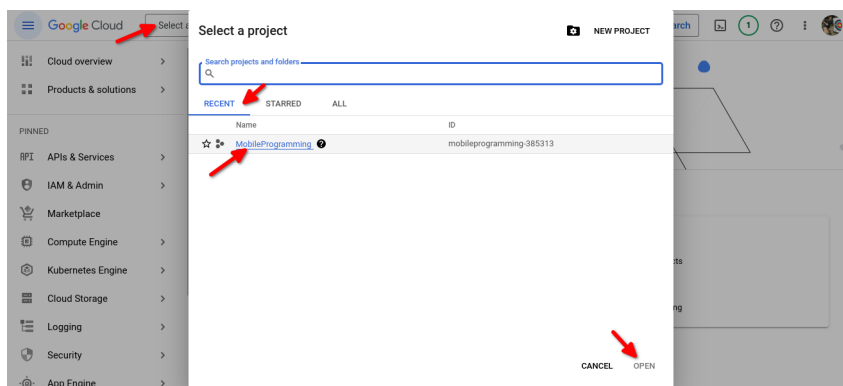
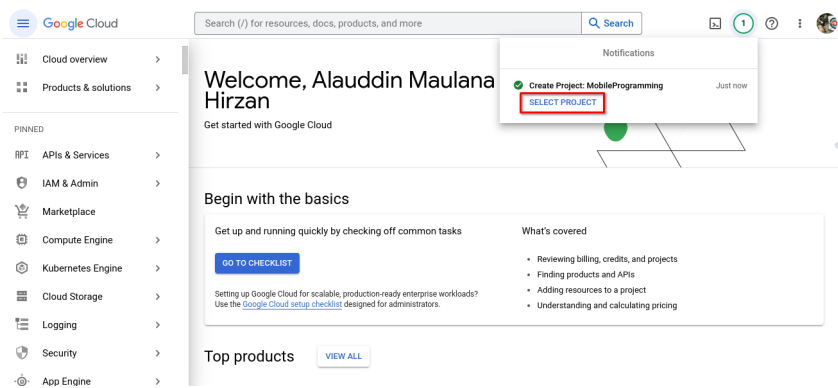
5. Buatlah projek baru dengan klik **Select Project** di bagian atas, dan pilih **New Project**



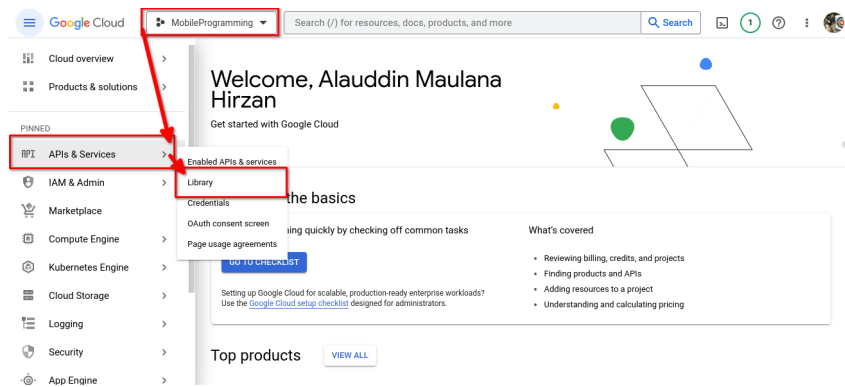
6. Beri nama projek **MobileProgramming** tanpa spasi, dan klik **Create**



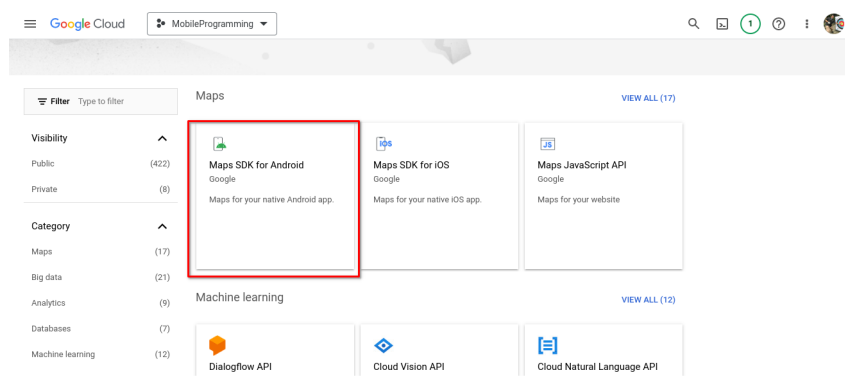
7. Tunggu Google membuat proyeknya, dan pastikan proyek yang sudah dibuat sudah terpilih. Klik **Select Project** di bagian notifikasi atau Dasbor



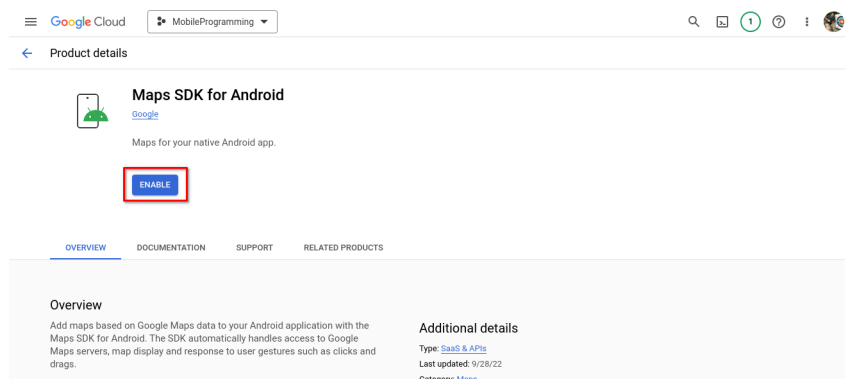
8. Berikutnya adalah membuka akses proyek ke GoogleApps Maps. Sesudah memilih proyek, di bagian panel kiri pilih **APIs & Services**, lalu pilih **Library**



9. Halaman akan dialihkan ke daftar Apps Google, pilih **Maps SDK for Android**



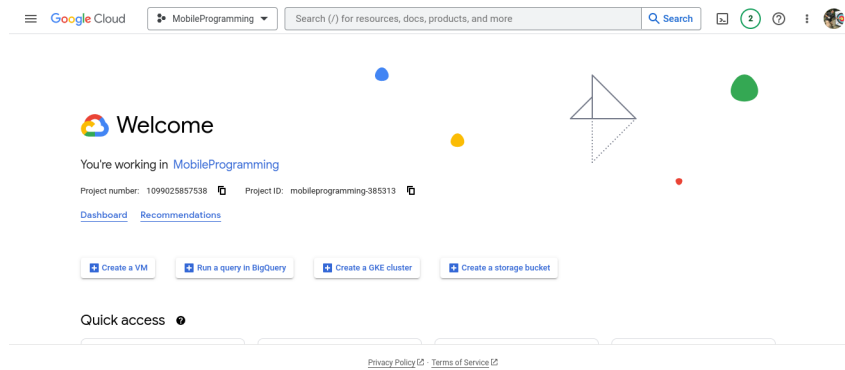
10. Klik **Enable** untuk mengizinkan proyek mengakses **Maps SDK for Android**



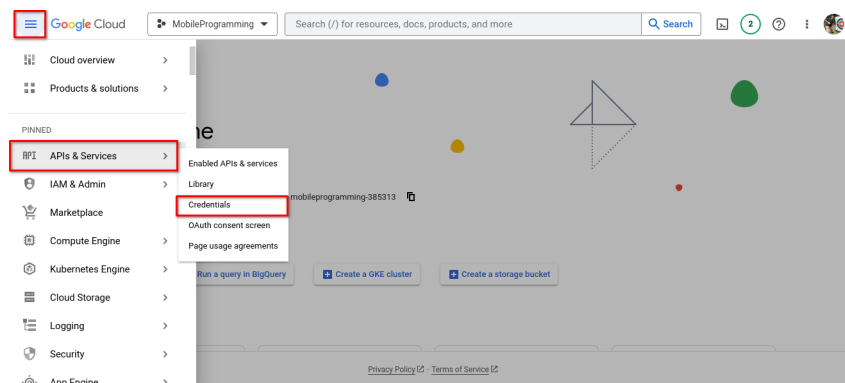
11. Tunggu beberapa saat hingga halaman berubah. Jika tidak refresh halaman.

12. Jika Google meminta pendaftaran pembayaran / **Billing** klik Cancel. Atau klik **Logo Google Cloud**

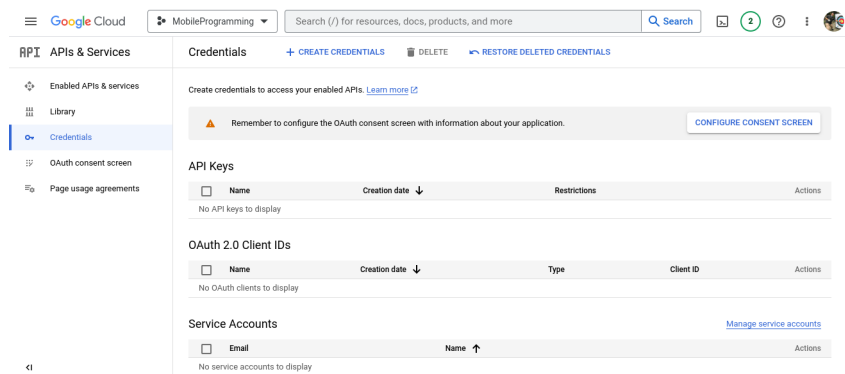
13. Kembali ke halaman Dasbor proyek



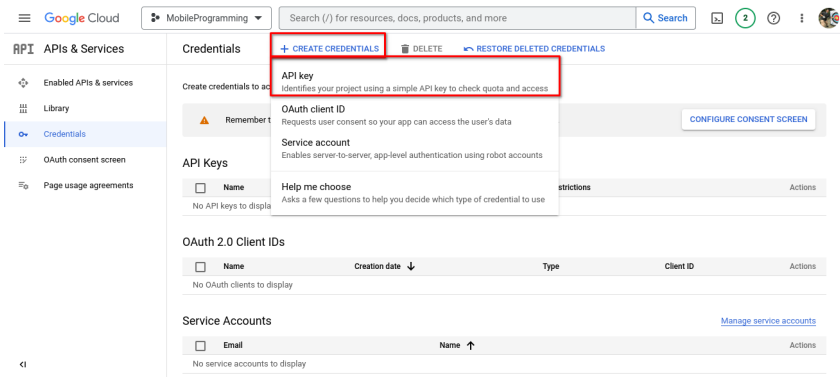
14. Klik **3 Garis Hitam** di pojok kiri atas, pilih **APIs & Services**, pilih **Credentials**



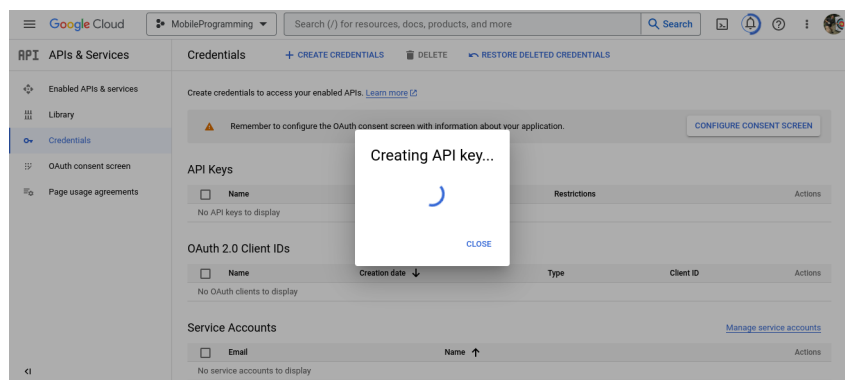
15. Halaman akan berpindah, dan di sinilah mahasiswa akan membuat API Keys untuk Android Studio



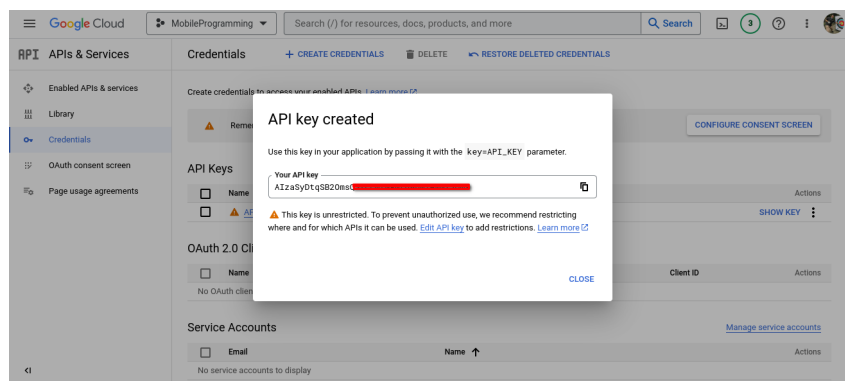
16. Untuk membuat API Keys, klik **Create Credentials**, pilih API Key



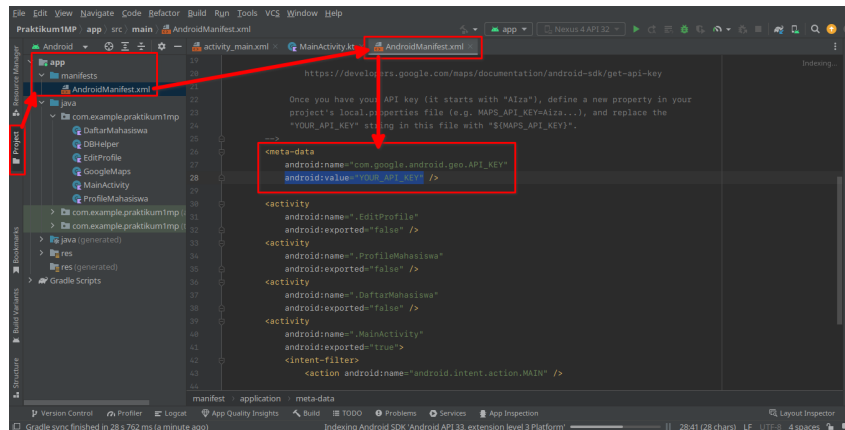
17. Google akan membuat kunci, tunggu hingga selesai



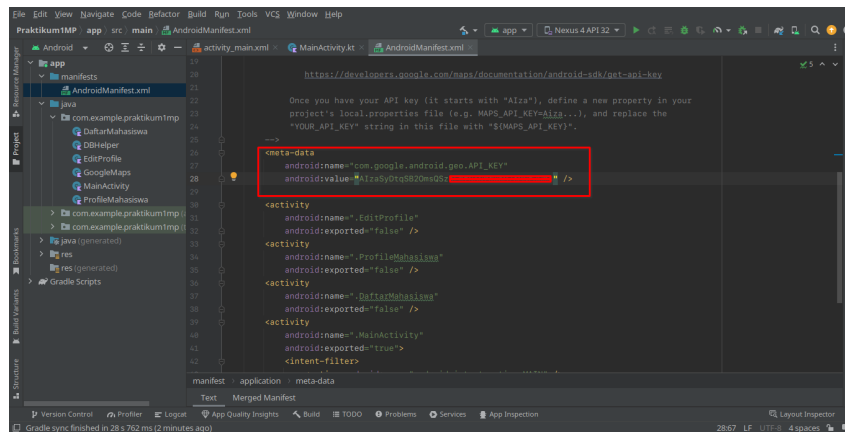
18. Kunci akan ditampilkan, dan klik **Copy**



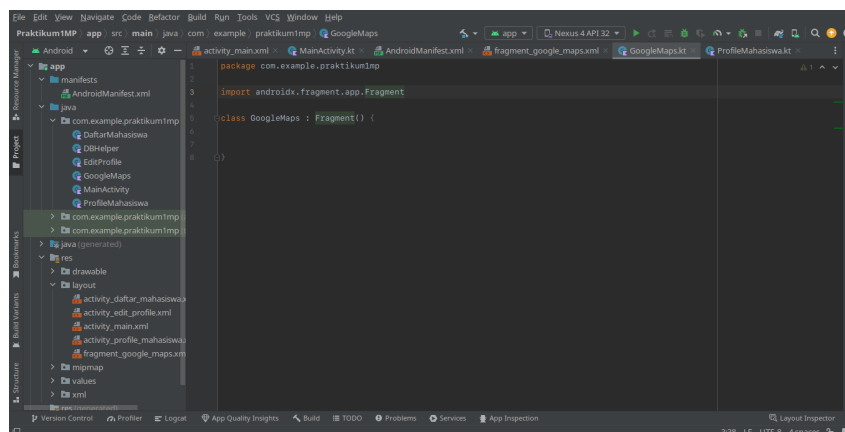
19. Buka Android Studio kembali, dan buka file **Android Manifest** atau klik **Project** → folder **manifests** → **AndroidManifest.xml**. Cari garis kode seperti di gambar berikut:



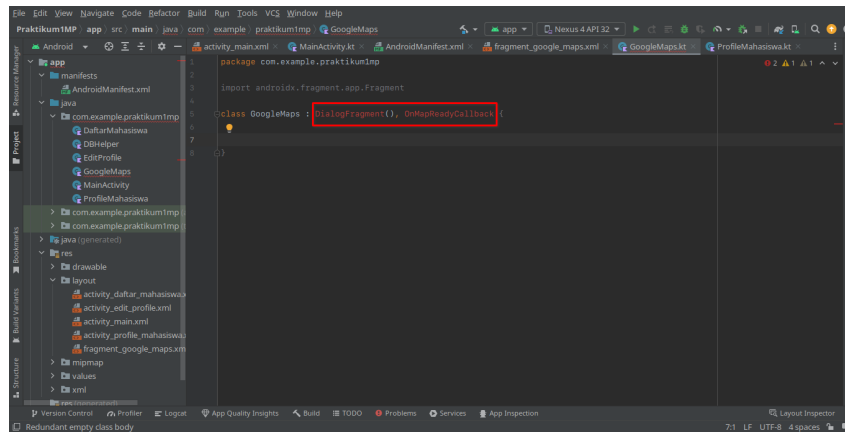
20. Tempel **API Key** yang sudah dibuat tadi menggantikan **YOUR_API_KEY** dalam tanda petik:



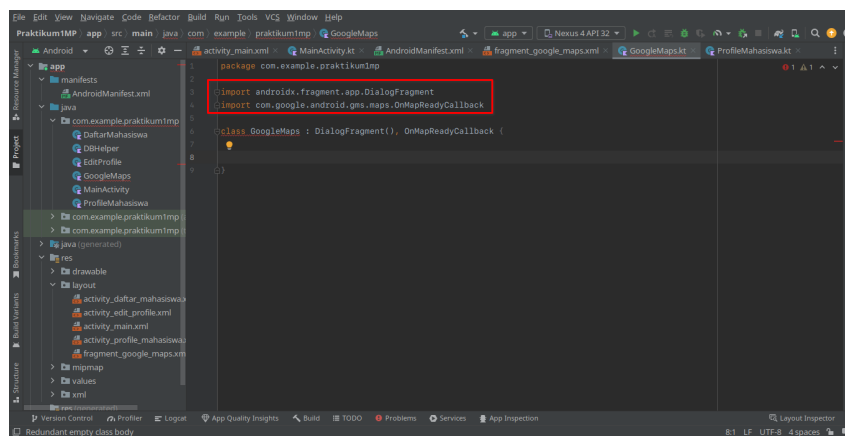
21. Berikutnya adalah menghapus semua isi dari **GoogleMaps.kt** dan hanya meninggalkan baris kode **Class** saja. Lihat gambar:



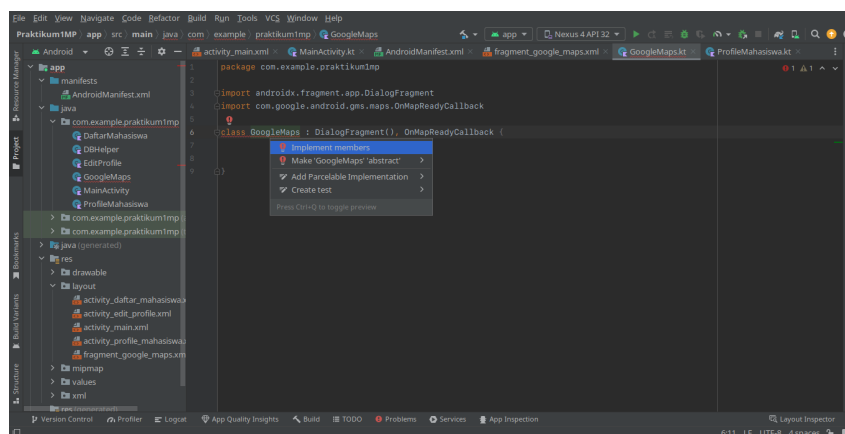
22. Ubah sintaks **Fragment** menjadi **DialogFragment()**, **OnMapReadyCallback**. Perhatikan contoh gambar:

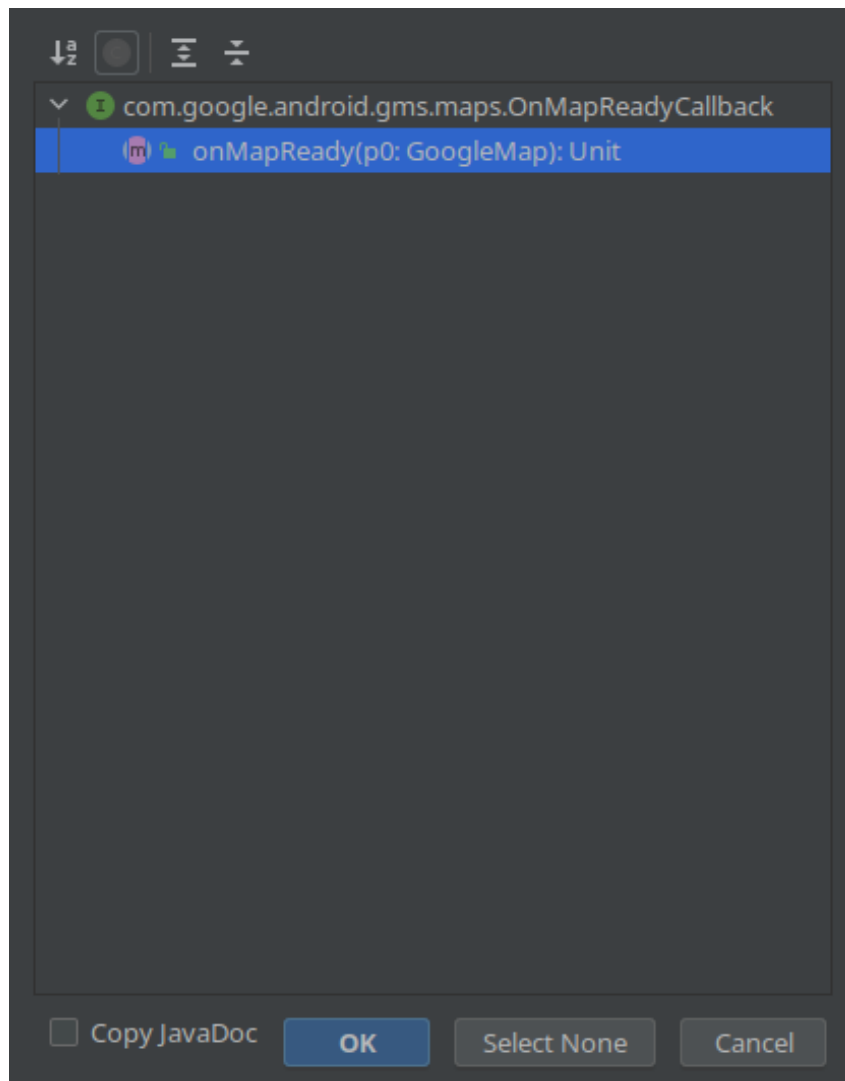


23. Perbaiki **Error** dengan **Import ALT+ENTER**



24. Sekarang baris kode **class GoogleMaps** menjadi **Error**. Klik kanan pilih **Show Context Actions**, pilih **Implement Members**, dan klik OK



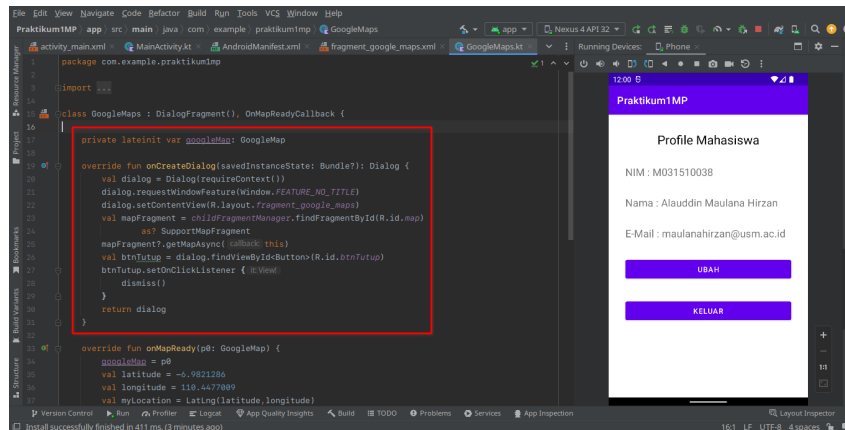


25. Berikutnya memrogram file tersebut, letakkan kode berikut ini tepat di bawah baris kode **class GoogleMaps**

Potongan Kode

```
private lateinit var googleMap: GoogleMap

override fun onCreateDialog(savedInstanceState: Bundle?): Dialog {
    val dialog = Dialog(requireContext())
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE)
    dialog setContentView(R.layout.fragment_map)
    val mapFragment = childFragmentManager.findFragmentById(R.id.map)
        as? SupportMapFragment
    mapFragment?.getMapAsync(this)
    val btnTutup = dialog.findViewById<Button>(R.id.btnTutup)
    btnTutup.setOnClickListener {
        dismiss()
    }
    return dialog
}
```

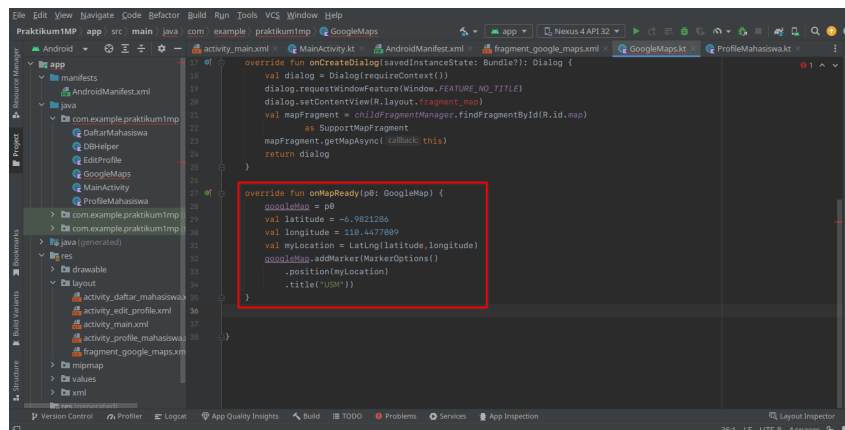



26. ABAIKAN ERROR R.layout.fragment_map. Lanjutkan kode dengan mengubah isi onMapReady dengan kode berikut

```

Potongan Kode
googleMap = p0
val latitude = -6.9821286
val longitude = 110.4477009
val myLocation = LatLng(latitude,longitude)
googleMap.addMarker(MarkerOptions()
    .position(myLocation)
    .title("USM"))
googleMap.uiSettings.isZoomControlsEnabled = true
googleMap.uiSettings.isMyLocationButtonEnabled = true
googleMap.addMarker(MarkerOptions().position(myLocation))
googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(myLocation, 15f))

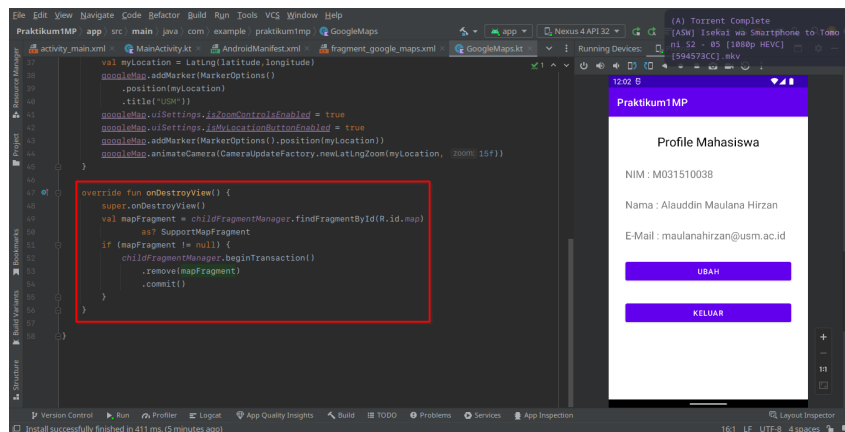
```



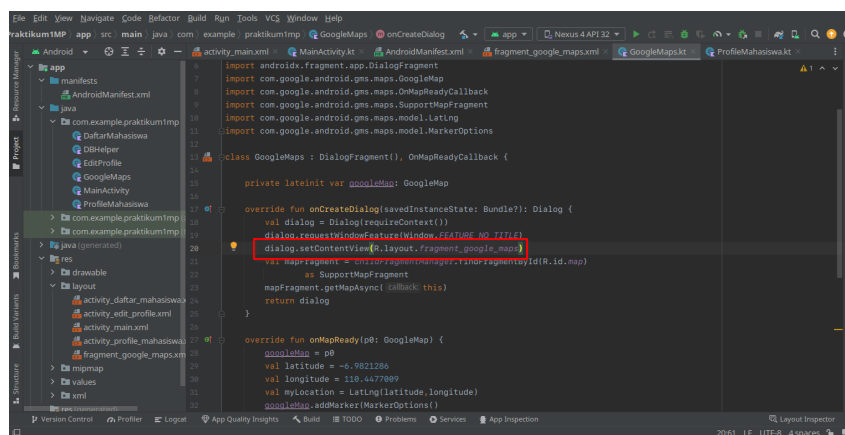
27. Bagian berikutnya, lanjutkan kode dengan kode berikut

Potongan Kode

```
override fun onDestroyView() {  
    super.onDestroyView()  
    val mapFragment = childFragmentManager.findFragmentById(R.id.map)  
    as? SupportMapFragment  
    if (mapFragment != null) {  
        childFragmentManager.beginTransaction()  
            .remove(mapFragment)  
            .commit()  
    }  
}
```



28. Untuk memperbaiki **Error fragment_map**. Ubah kode tersebut menjadi **fragment_google_maps**



29. Berikutnya ubah isi file **fragment_google_maps.xml** dengan membungkus kode yang ada ke dalam **LinearLayout**. Perhatikan kode dan Gambar:

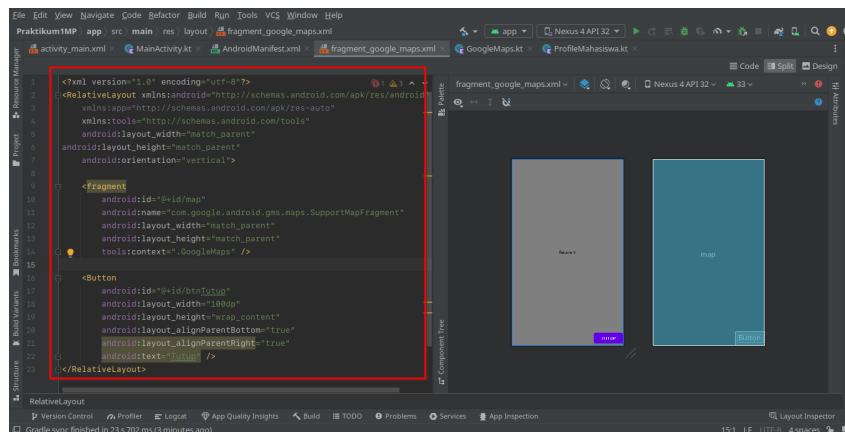
Potongan Kode

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

-- Kode Fragment Asli --

```
<Button
    android:id="@+id/btnTutup"
    android:layout_width="100dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:text="Tutup" />
```

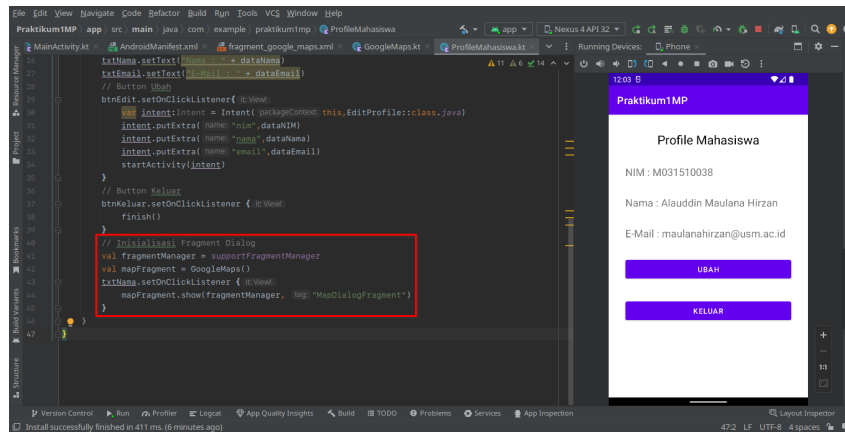
```
</RelativeLayout>
```



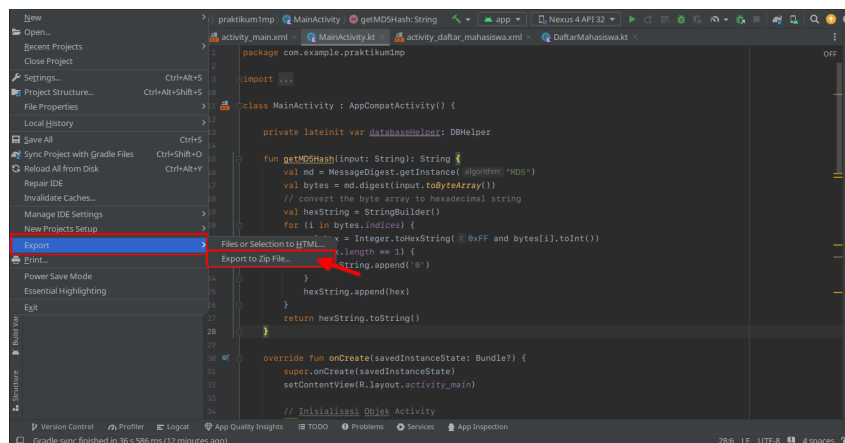
30. Jika sudah, buka file **ProfileMahasiswa.kt** dan tambahkan kode berikut tepat di bawah kode **btnKeluar**. Perhatikan kode dan gambar:

Potongan Kode

```
// Inisialisasi Fragment Dialog
val fragmentManager = supportFragmentManager
val mapFragment = GoogleMaps()
txtNama.setOnClickListener {
    mapFragment.show(fragmentManager, "MapDialogFragment")
}
```



31. Jalankan Emulator dan Aplikasi. Saat ini aplikasi hanya bisa membuka Map tanpa Zoom Auto dan Marker, dan Aplikasi akan Error ketika membuka map kedua kalinya
32. Untuk mengirimkan hasilnya, klik **File**, pilih **Export**, dan pilih **Export to ZIP**.



33. Pilih lokasi yang mudah diingat, pastikan nama dan format ZIP sudah sesuai, dan klik **OK**

