



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 Mengetahui Android	2
0.2 Mengetahui Android Studio	2
0.3 Mengetahui Android SDK	3
0.4 Mengetahui Android Emulator	3
Persiapan Praktikum	5
0.5 Perangkat Keras	5
0.6 Perangkat Lunak	5
1 Praktikum 1	6
1.1 Instalasi Android Studio	6
1.2 Tutorial	6
2 Praktikum 2	12
2.1 Konfigurasi SDK Android Studio dan USB Debugging	12
2.2 Tutorial	12
3 Praktikum 2	26
3.1 Antarmuka Halaman Login	26
3.2 Tutorial	26
4 Praktikum 4	37
4.1 Pemrograman Halaman Login	37
4.2 Tutorial	37
5 Praktikum 5	45
5.1 Pemrograman Halaman Pendaftaran	45
5.2 Tutorial	45
6 Praktikum 6	53
6.1 SQLite Insert Data	53
6.2 Tutorial	53
7 Praktikum 7	60
7.1 SQLite Read Data	60
7.2 Tutorial	60

8	Praktikum 8	69
8.1	SQLite Update Data	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

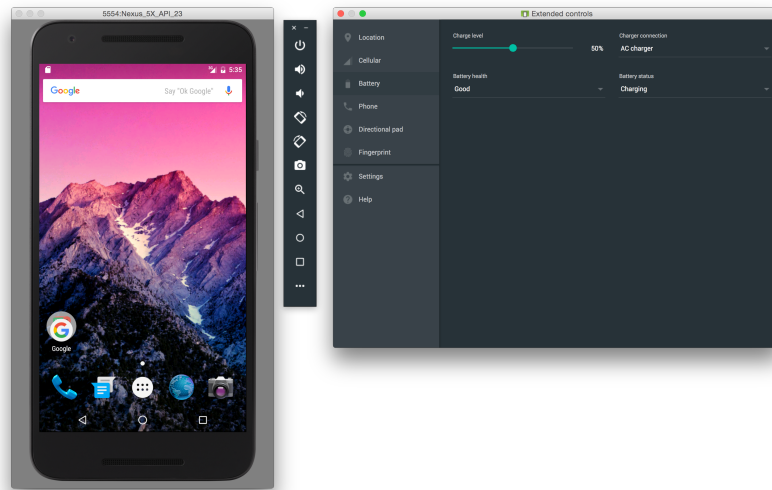
Merupakan framework dasar untuk membuat aplikasi Android dan sudah tersedia secara default di Android Studio. Android SDK memiliki berbagai macam versi tergantung dari versi Android itu sendiri. Penggunaan Android SDK juga akan mempengaruhi kompatibilitas ke belakang dengan Android versi lama.



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

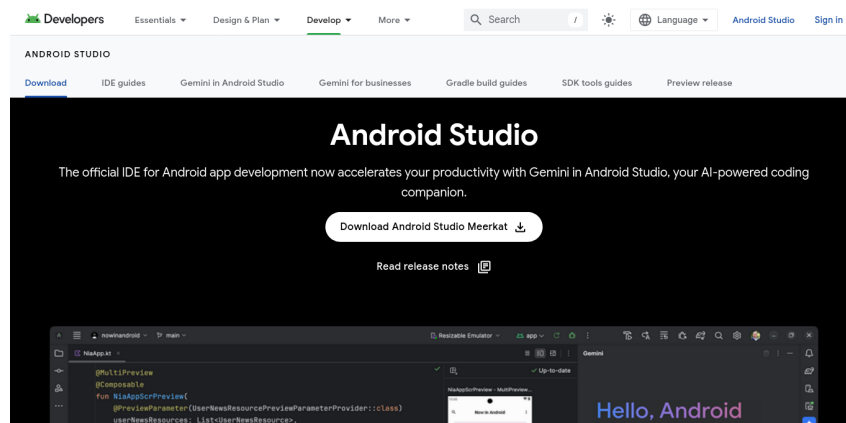
Praktikum 1

1.1 Instalasi Android Studio

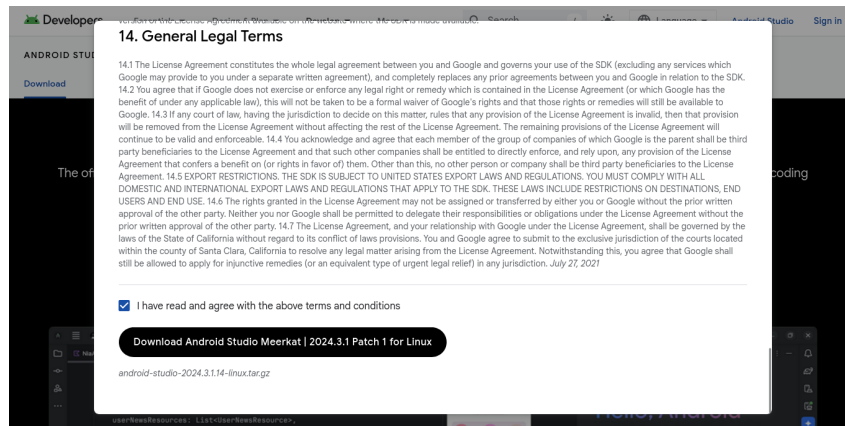
Di bagian ini mahasiswa diajarkan bagaimana mengunduh Android Studio, lalu menginstall IDE di Komputer yang ada. Hal ini sangat direkomendasikan agar mahasiswa dapat memastikan Android Studio dapat berjalan baik di Komputer

1.2 Tutorial

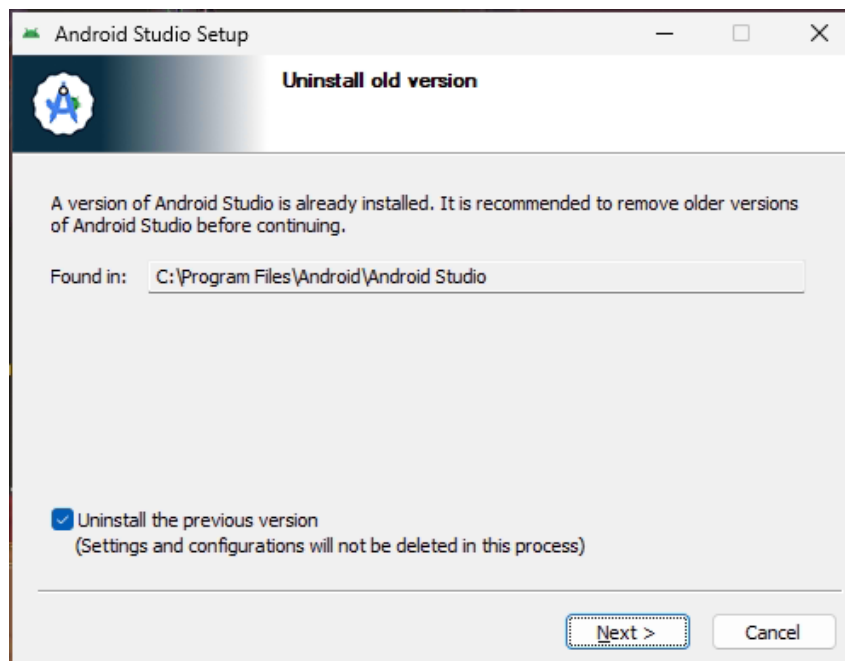
1. Mahasiswa wajib mengunduh Android IDE terbaru untuk memastikan versi yang digunakan semua mahasiswa adalah sama dan terbaru. Link : [Klik Link](#)



2. Centang **I have read and agree with the above terms and conditions**. Lalu Klik **Download Android Studio**



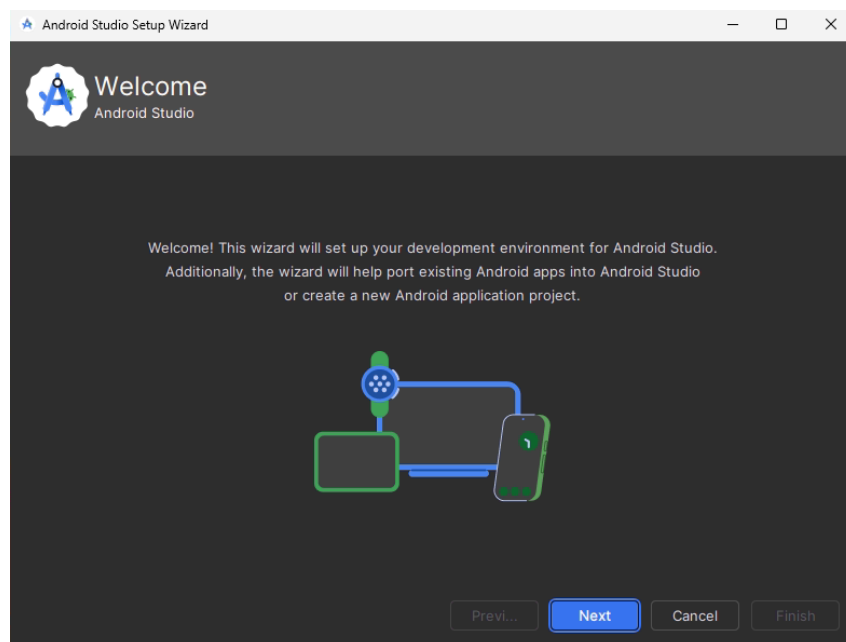
3. Setelah selesai menginstall, mahasiswa melakukan instalasi **Android Studio** tersebut. Jika diminta untuk menghapus versi lama, lakukan penghapusan tersebut. Jika tidak, lanjut tahap berikutnya



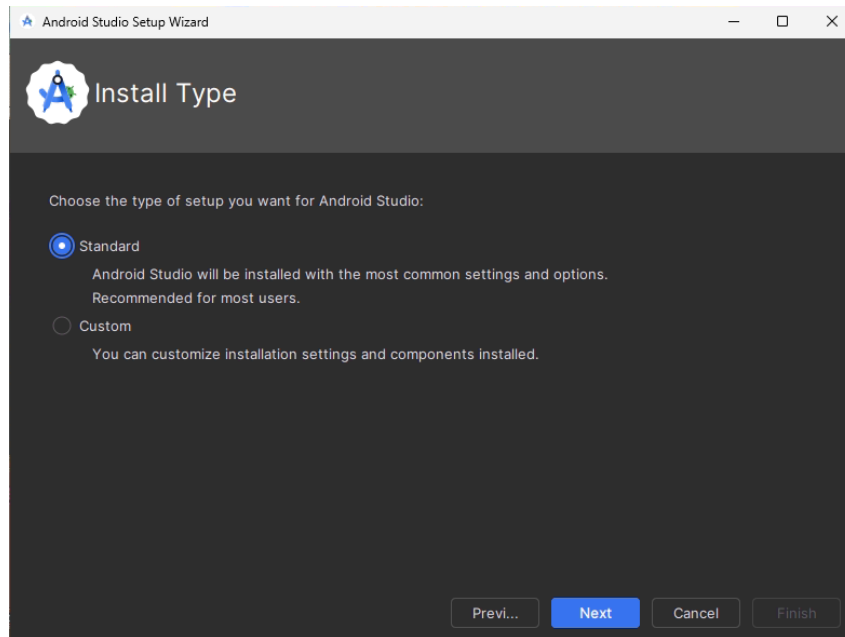
4. Klik Next hingga proses Instalasi selesai



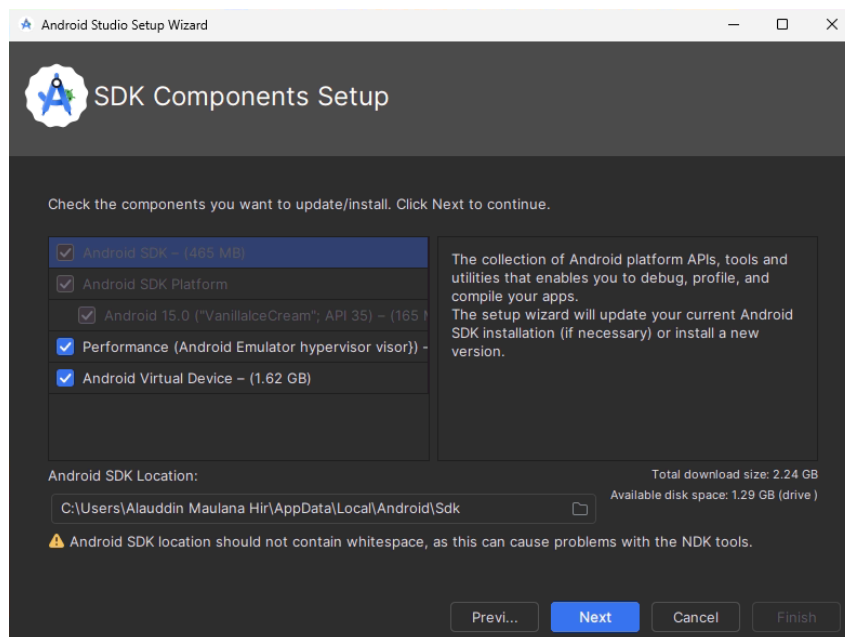
5. Setelah selesai, Android Studio akan secara otomatis menjalankan Android Studio. Klik Next di bagian Welcome



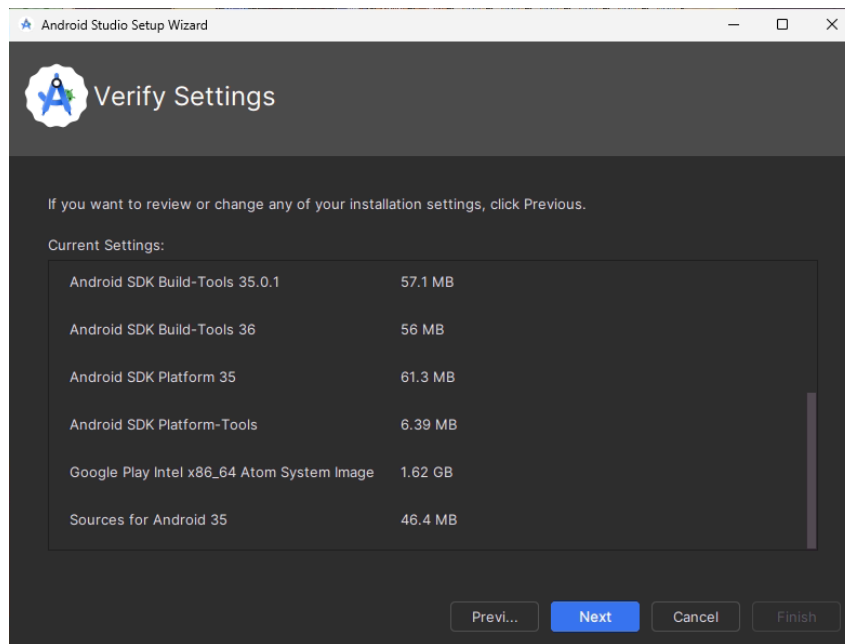
6. Pilih **Standard**, lalu klik **Next**



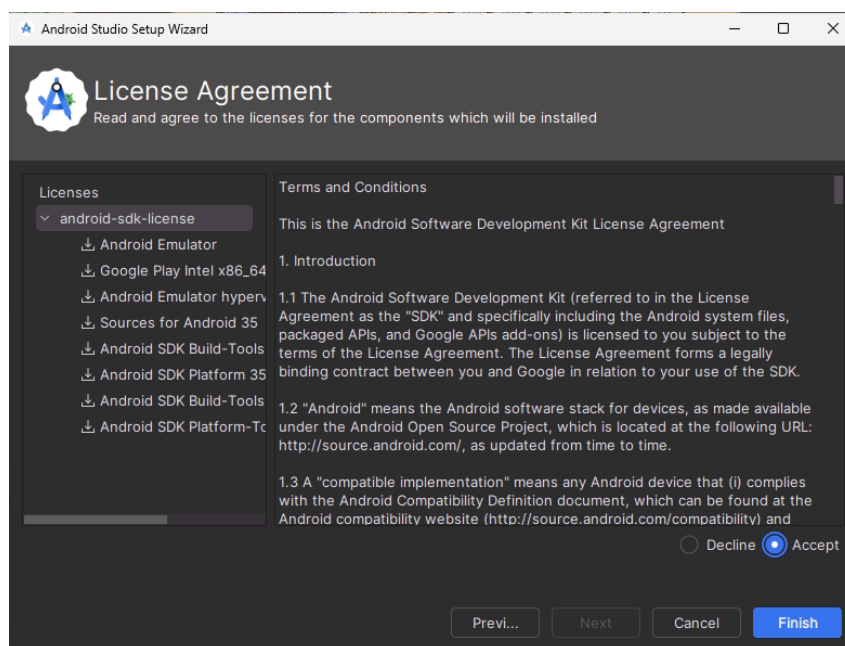
7. Pastikan semua tercentang lalu klik **Next** lagi



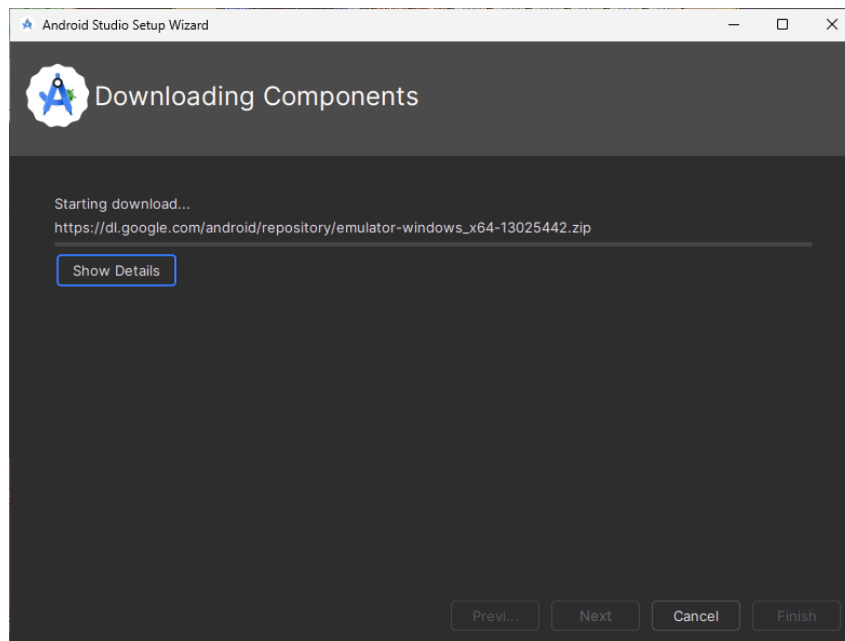
8. Verifikasi semuanya, lalu klik **Next** lagi



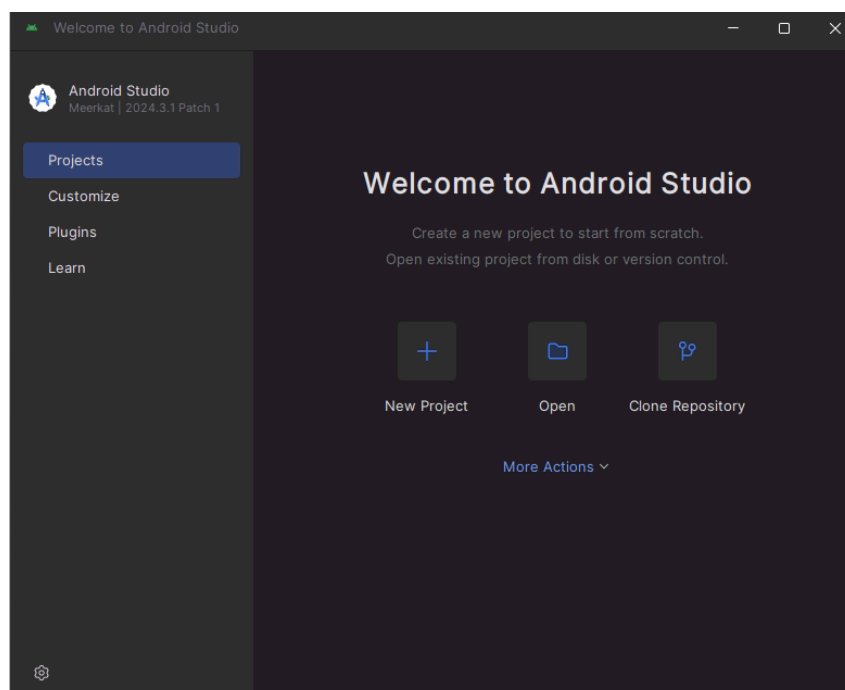
9. Di bagian **License Agreement**, pilih **Accept** lalu **Finish**



10. Android Studio akan mengunduh file-file yang diperlukan. Proses ini memakan waktu tergantung koneksi internet.



11. Android Studio sudah terpasang dan siap digunakan



Bab 2

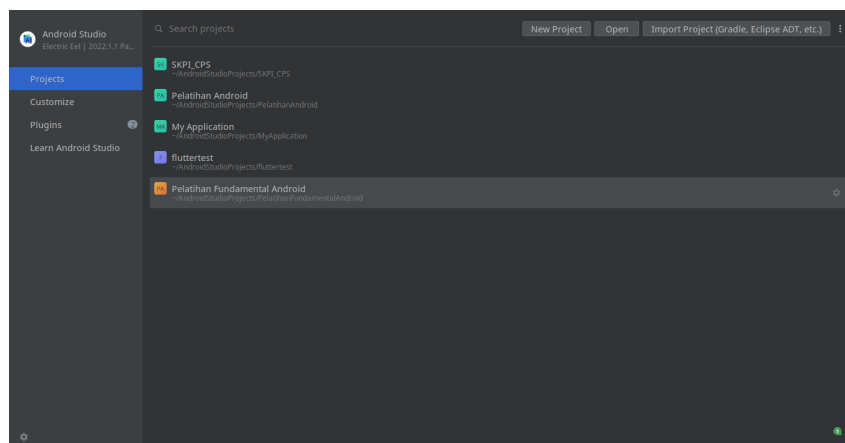
Praktikum 2

2.1 Konfigurasi SDK Android Studio dan USB Debugging

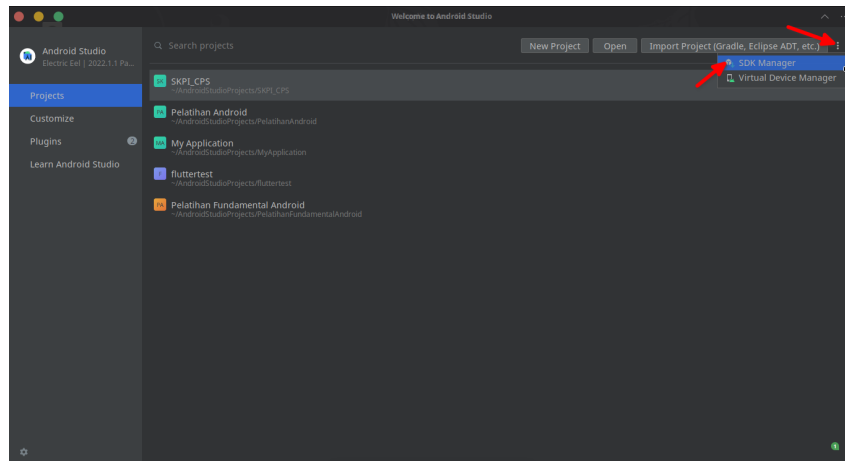
Di bagian ini mahasiswa diajarkan bagaimana mengkonfigurasi Android Studio khususnya pada bagian Android SDK, dan Android Emulator di komputer masing-masing. Hal ini sangat direkomendasikan agar mahasiswa dapat mengenali struktur Android Studio dengan lebih mudah,

2.2 Tutorial

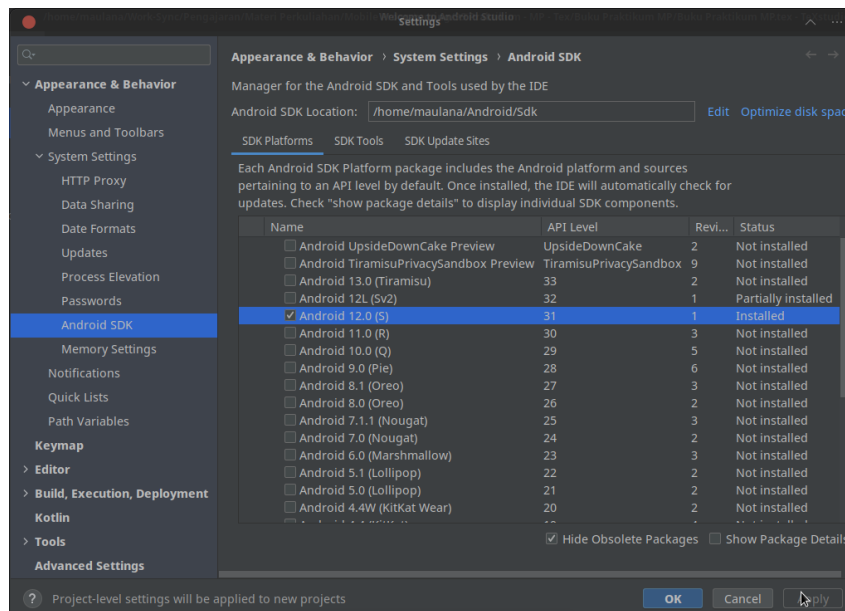
1. Buka **Android Studio**, dan pastikan menggunakan **Versi Terbaru** untuk memulai praktikum ini.



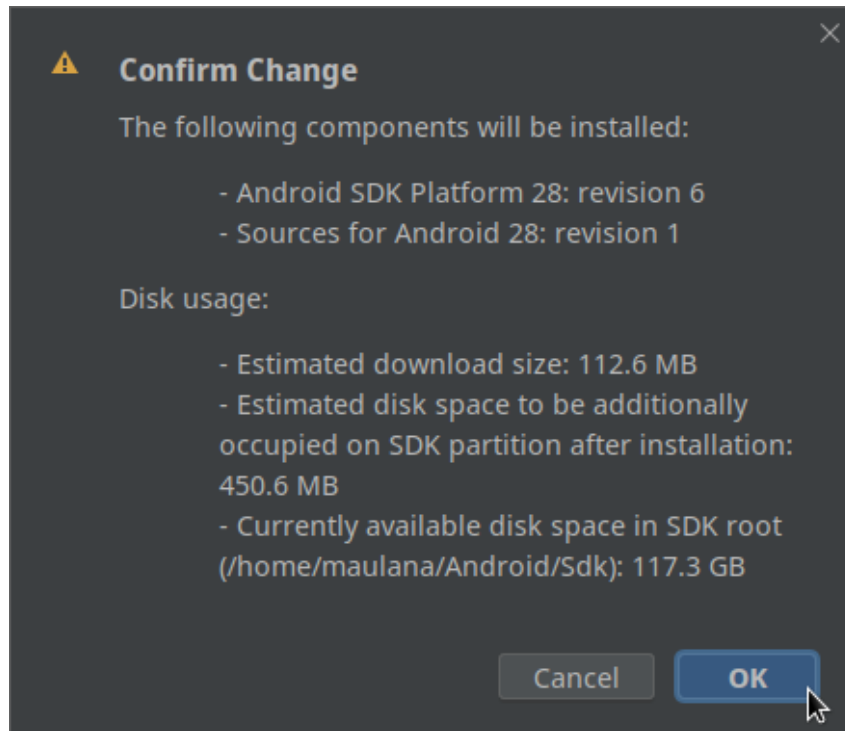
2. Untuk mengkonfigurasi Android SDK, klik **Titik Tiga / More Action** → **SDK Manager**



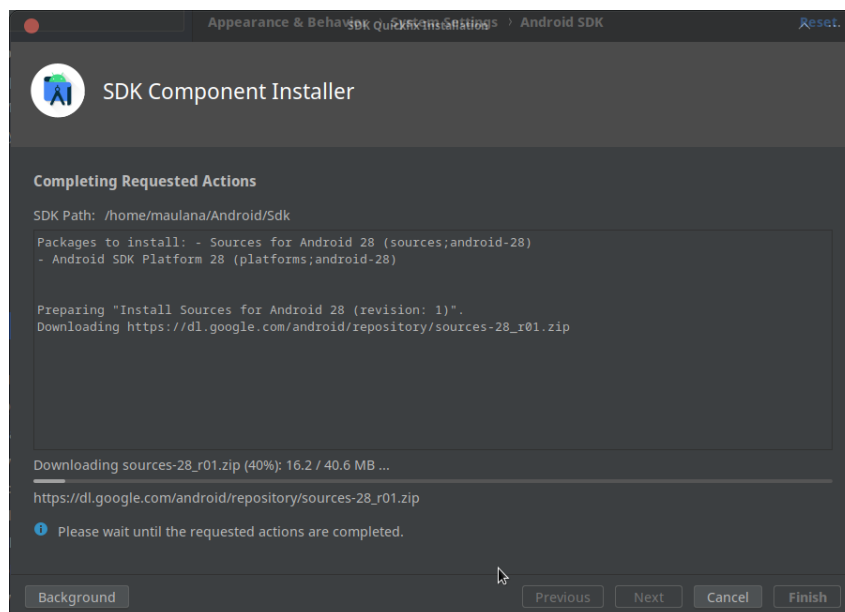
3. Jika sudah muncul tampilan **SDK Manager**, cek apakah ada SDK yang sudah terinstall. Secara **Default**, SDK sudah terpasang secara otomatis bersamaan dengan instalasi Android SDK. Langkah ini bisa dilewati



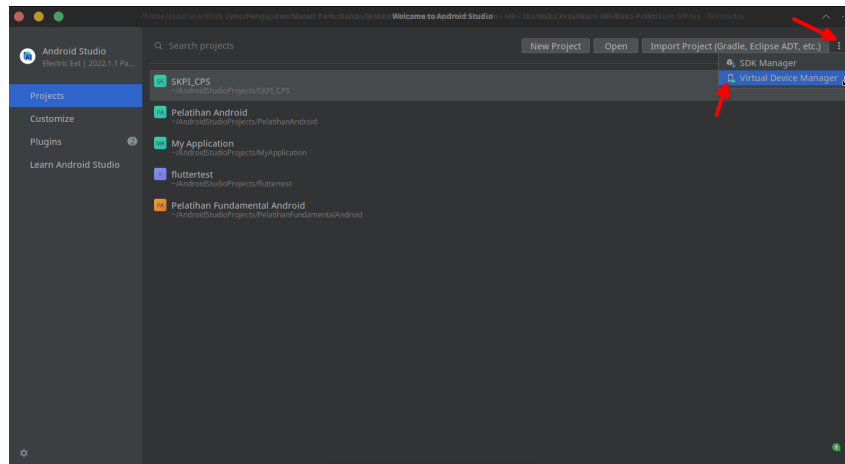
4. Untuk menginstall SDK secara manual, cukup centang SDK target dan klik **Apply**. Android Studio akan memunculkan peringatan (jika ada, klik OK)



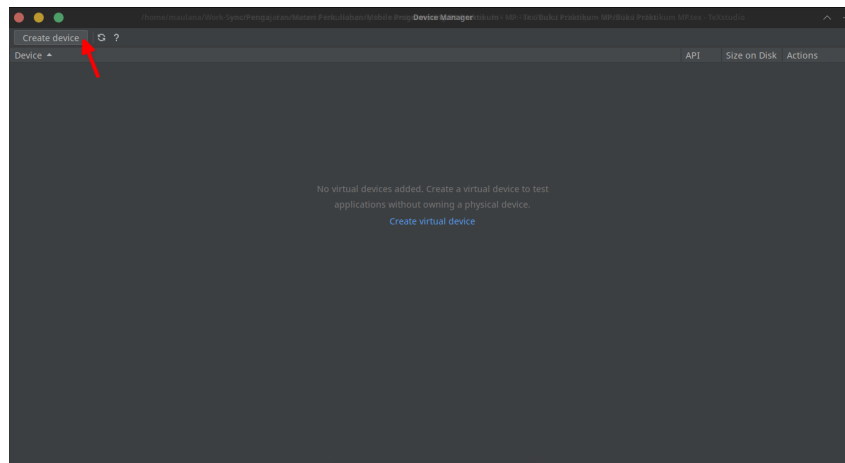
5. Kemudian proses instalasi SDK akan dimulai. Tunggu hingga selesai dan klik **Finish** untuk menutup **SDK Component Installer**



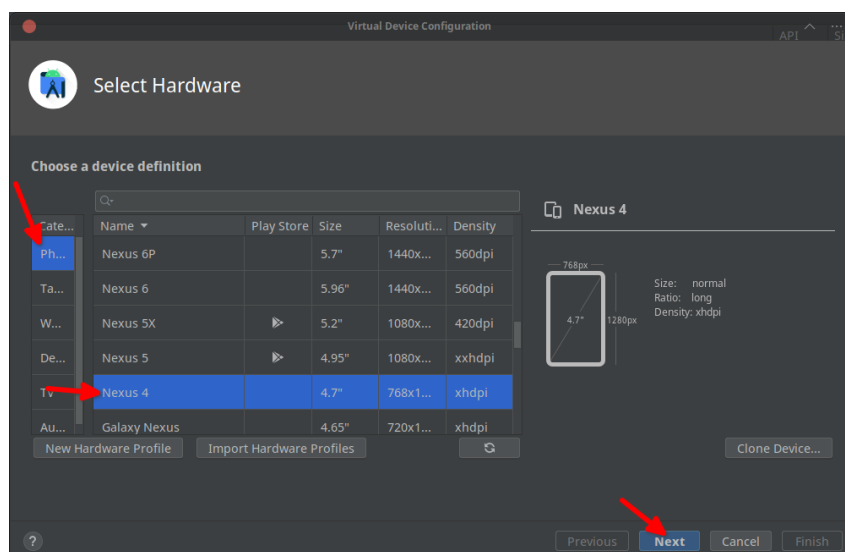
6. Lalu klik **OK** untuk menutup **SDK Manager**
7. Android Studio akan kembali menampilkan halaman **Welcome to Android Studio**. Kemudian konfigurasi perangkat virtualisasi dengan klik **Titik Tiga / More Action** → **Virtual Device Manager**



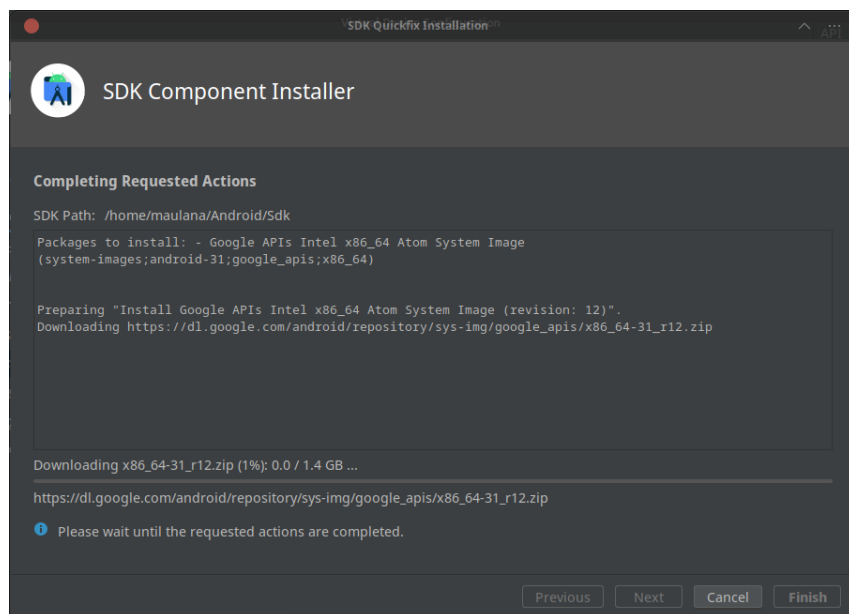
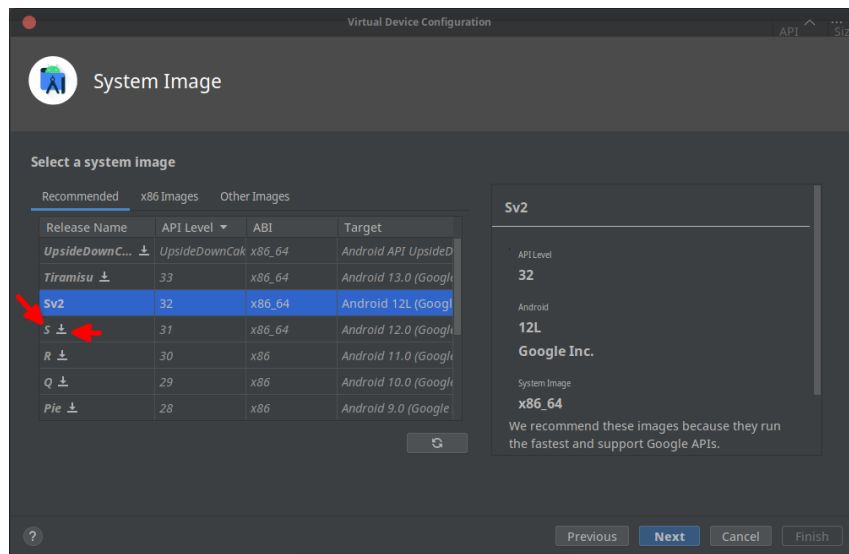
8. Berikutnya akan memunculkan konfigurasi perangkat Virtual. Secara **Default** halaman ini masih kosong, dan bisa memulai membuat dengan meng-klik **Create device**



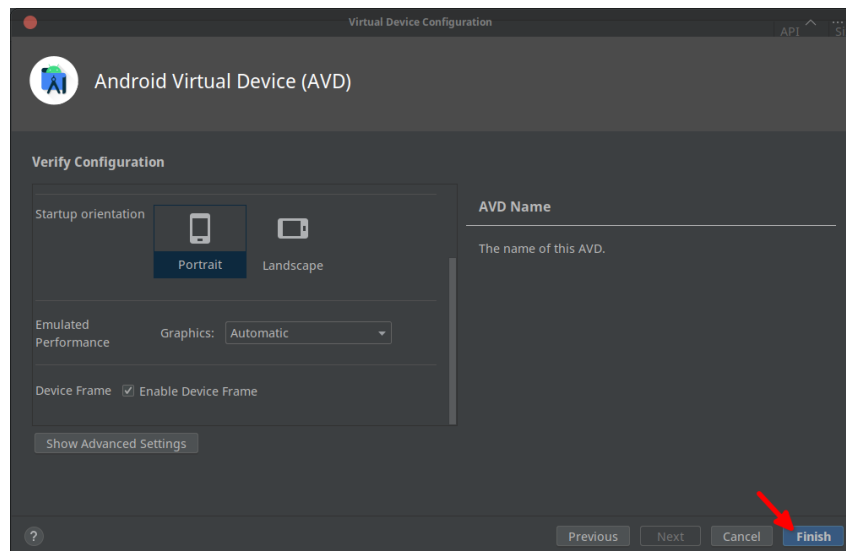
9. Android Studio akan memulai *Wizard* untuk membuat perangkat **Virtual Android**. Pilih **Phone** → **Nexus 4** → **Next**



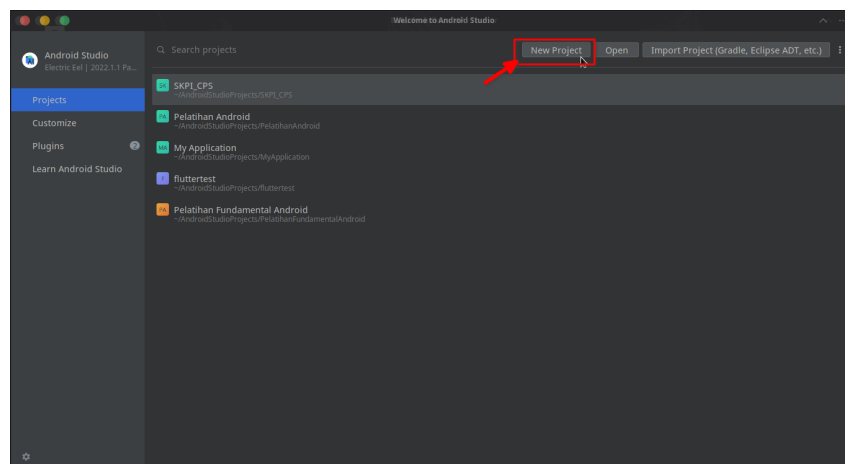
10. Pilih **Versi Android**, → Klik **Icon Download** di samping versi apabila belum terinstall → Klik **Next** apabila **System Image** sudah terinstall.



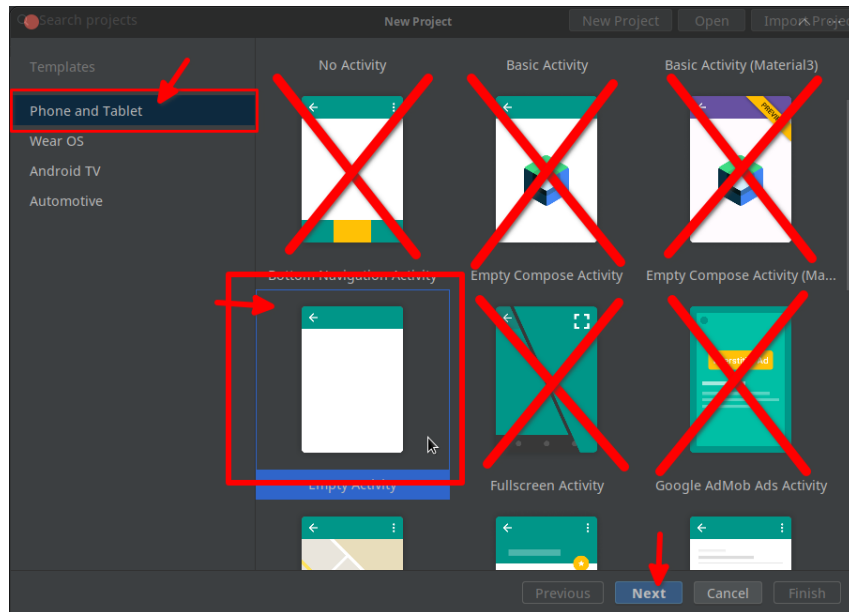
11. Berikutnya adalah konfigurasi perangkat itu sendiri, secara **Default** tidak perlu dirubah kecuali memahami konfigurasi lanjutan dari **Virtual Machine**. Klik **Finish**



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**.

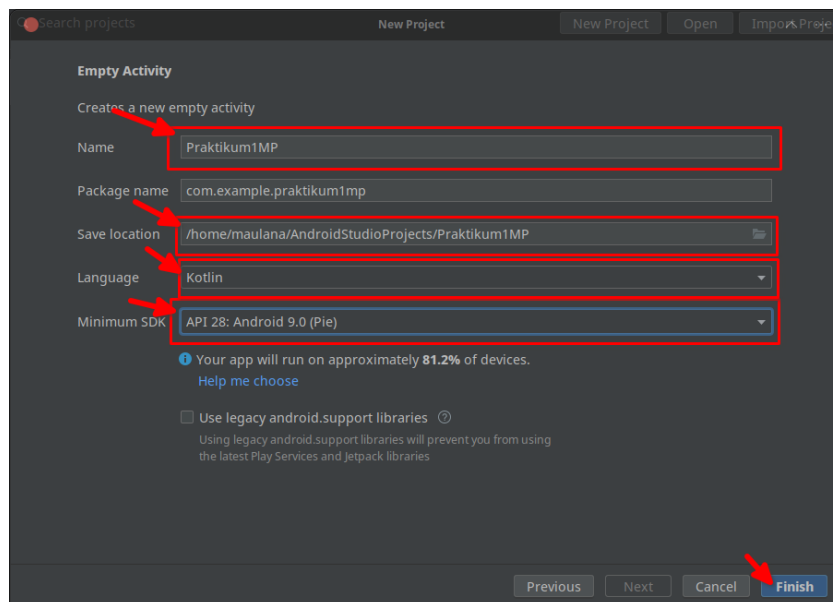


14. Android Studio akan membuka memperlihatkan konfigurasi proyek yang akan dibuat. Pilih **Phone and Tablet** → Klik **Empty Views Activity** => Klik **Next**

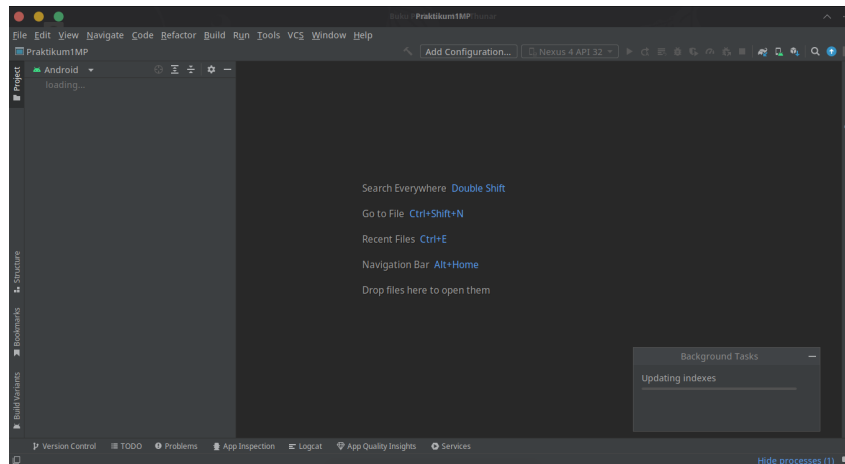


15. Sesuaikan konfigurasi proyek seperti berikut, dan klik **Finish**:

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

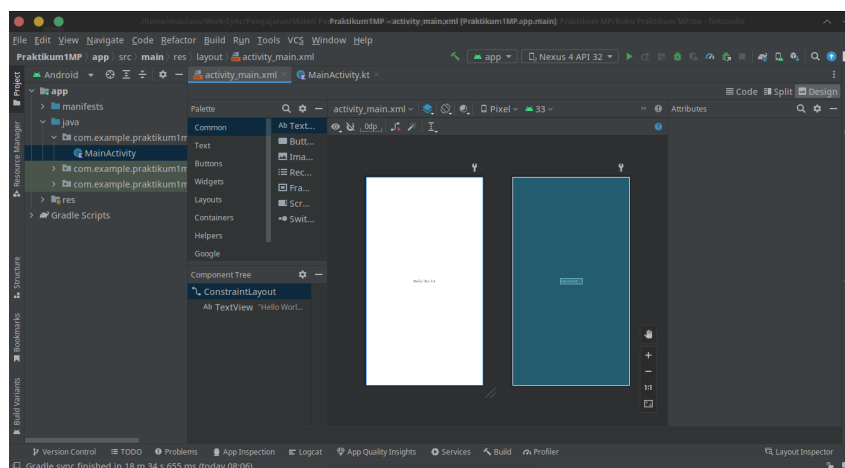
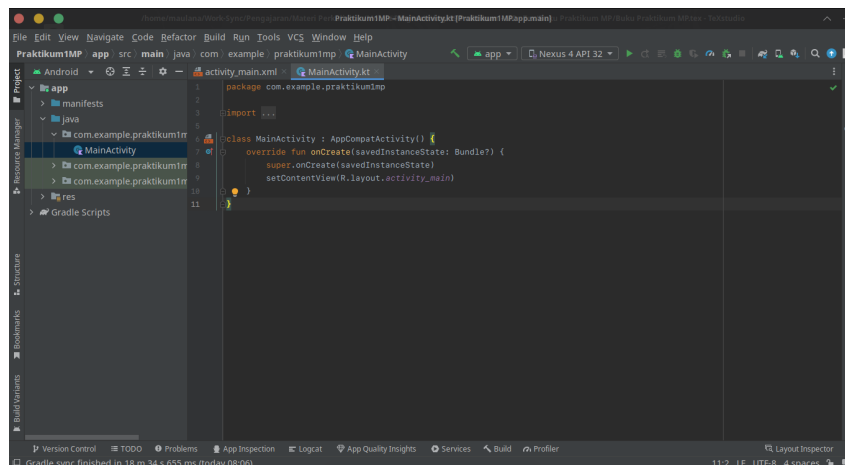


16. Android Studio akan menyiapkan file proyek, tunggu hingga selesai. Android Studio akan memperlihatkan kode-kode dan **Project Tree** di samping.

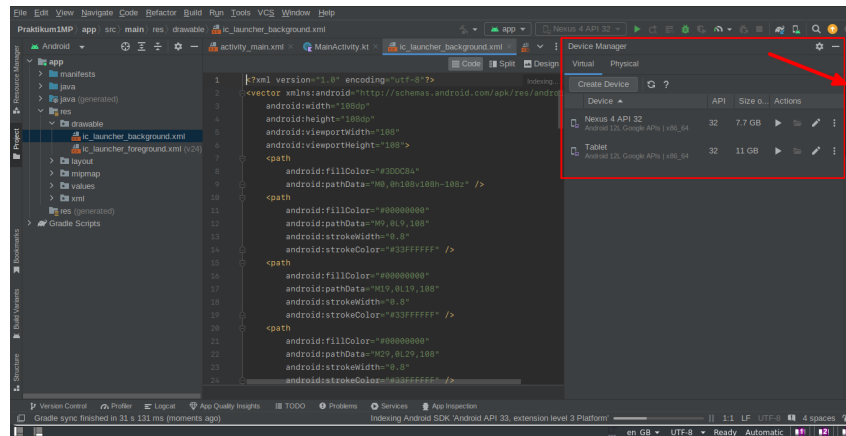


17. Android Studio akan membuat dua (2) file secara **default**:

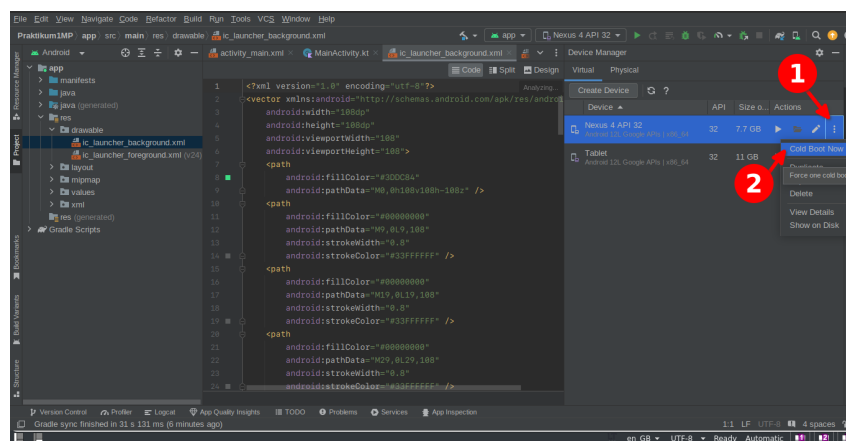
- **MainActivity.kt** : Digunakan untuk menaruh kode-kode fungsional menggunakan bahasa **Java/Kotlin**
- **activity_main.xml** : Digunakan untuk meletakkan objek-objek antarmuka aplikasi Android



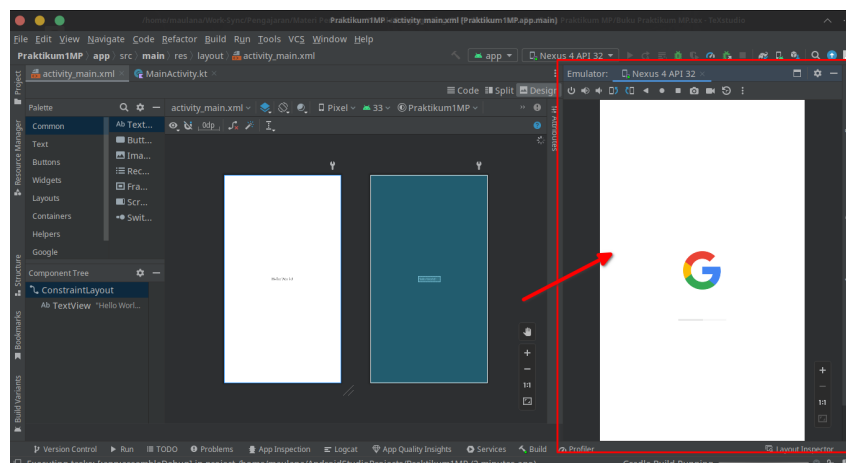
18. Android Studio sudah siap untuk membuat aplikasi Android. Namun sebelum membuat aplikasi, berikutnya adalah mencoba untuk menjalankan aplikasi **Hello World** ini ke **Virtual Device**
19. Klik **Device Manager** yang ada di bagian samping **Android Studio**



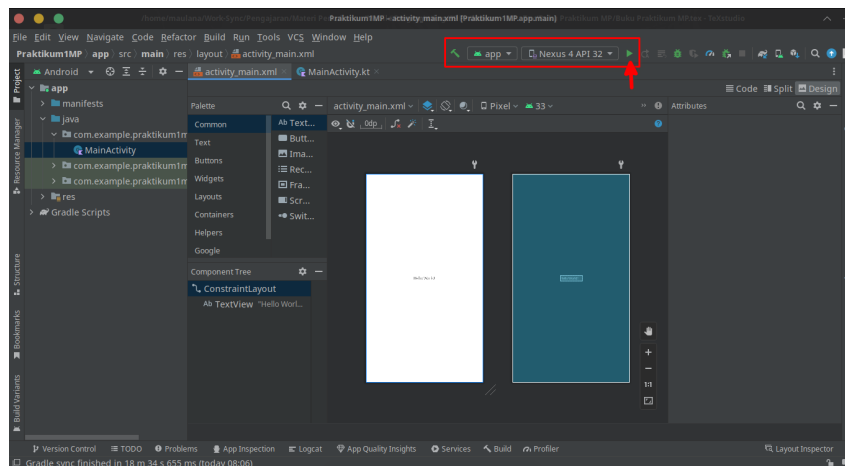
20. Tekan **Tiga Titik** yang ada di sebelah **Virtual Device**, pilih **Cold Boot Now**. Lakukan ini setiap kali memulai Praktikum



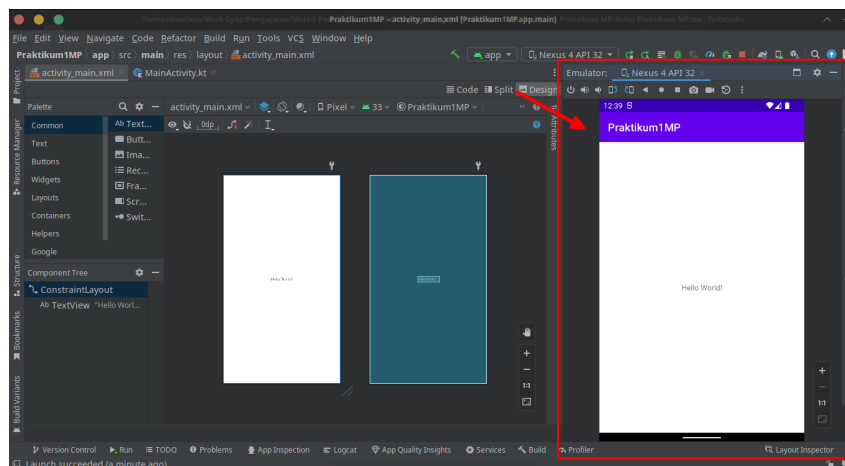
21. Jika Booting sudah selesai, tunggu proses Booting



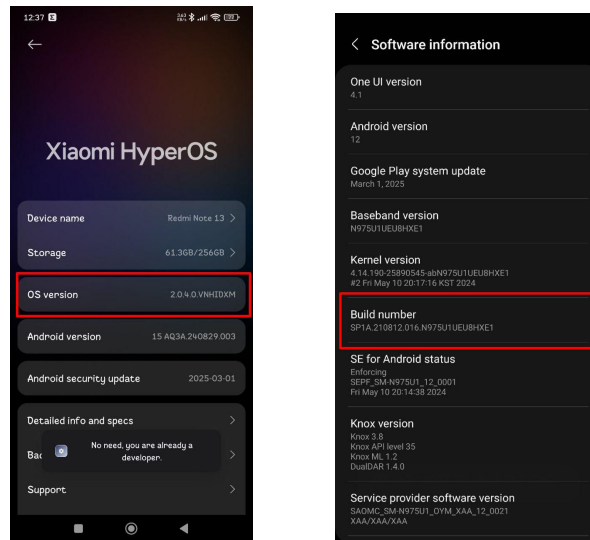
22. Jika sudah, jalankan aplikasi dengan klik tombol **Play**



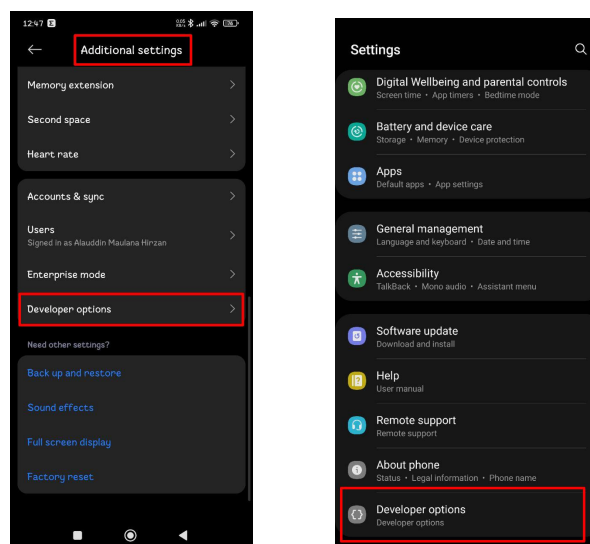
23. Jika sukses, maka Android Studio akan menampilkan aplikasi seperti gambar berikut



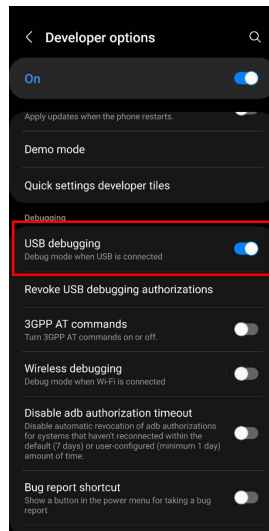
24. Untuk menghubungkan Smartphone Android ke Android Studio, pertama-tama pastikan sudah menyalakan **Opsi Pengembang/Developer Mode**. Dengan cara menekan **Build Number/OS Version** yang ada di **Tentang Telepon/About Phone**. Setiap HP memiliki tempat yang berbeda.



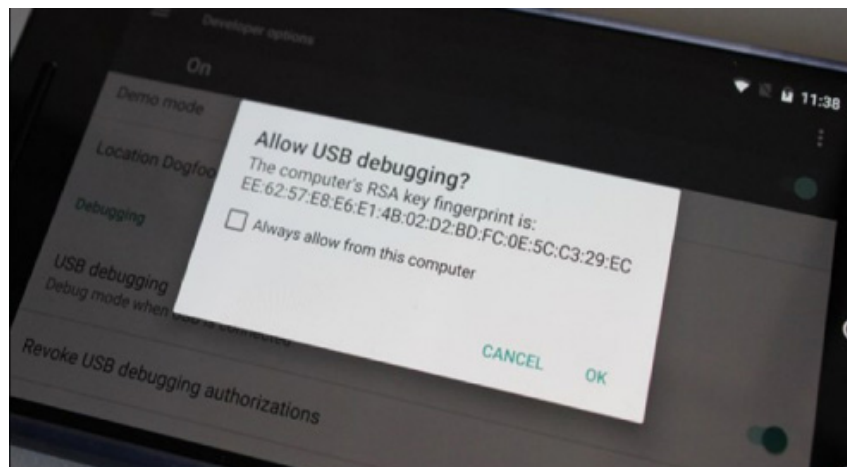
25. Cari menu **Opsi Pengembang/Developer Mode**. Menu ini berbeda letaknya sesuai perangkat smartphone



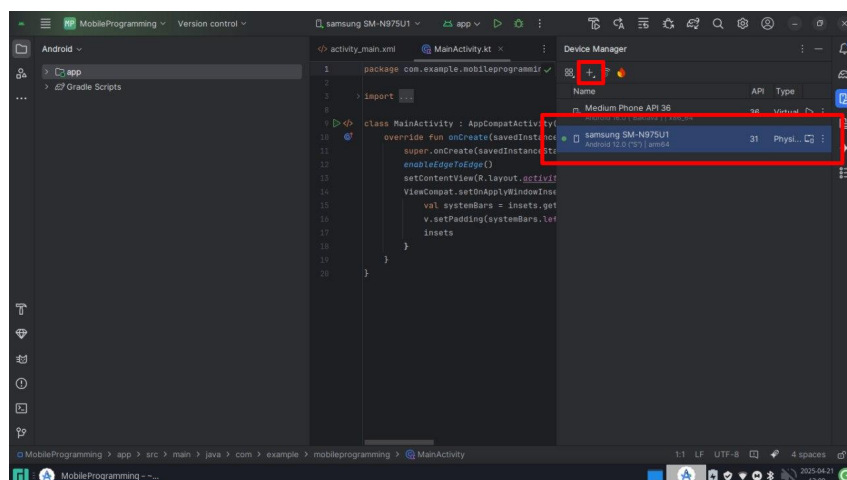
26. Menu untuk **Developer Options/Opsi Pengembang** relatif sama. Cari opsi **USB Debugging Mode** dan **Centang**



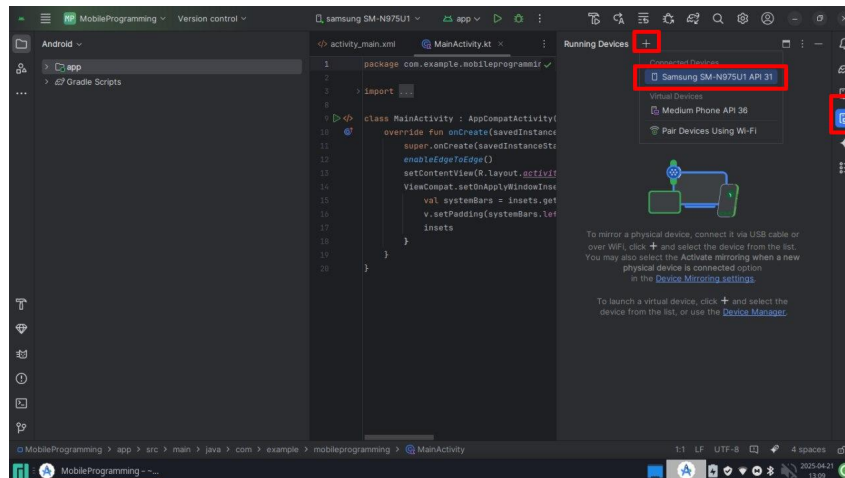
27. Gunakan **Kabel Data** untuk menghubungkan Smartphone dengan Komputer/Lap-top. Jika ada konfirmasi untuk mempercayai perangkat, **Centang** lalu pilih **Allow**



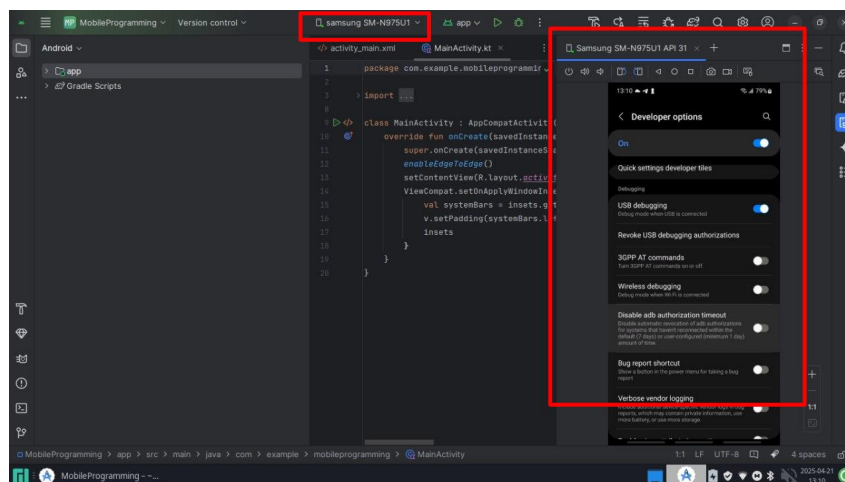
28. Jika sudah sukses, maka buka Android Studio. Cari **Device Manager** di panel kanan. Klik tanda Plus



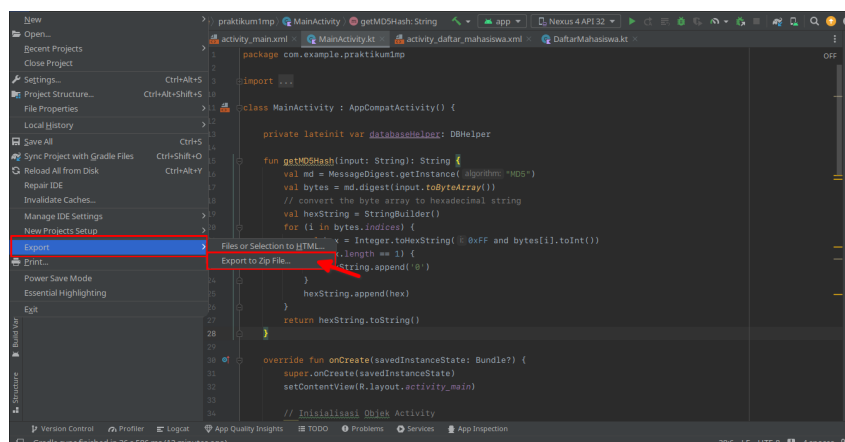
29. jika sudah muncul, pilih **Running Devices** di panel kanan, lalu klik Plus lagi dan tambahkan Perangkat



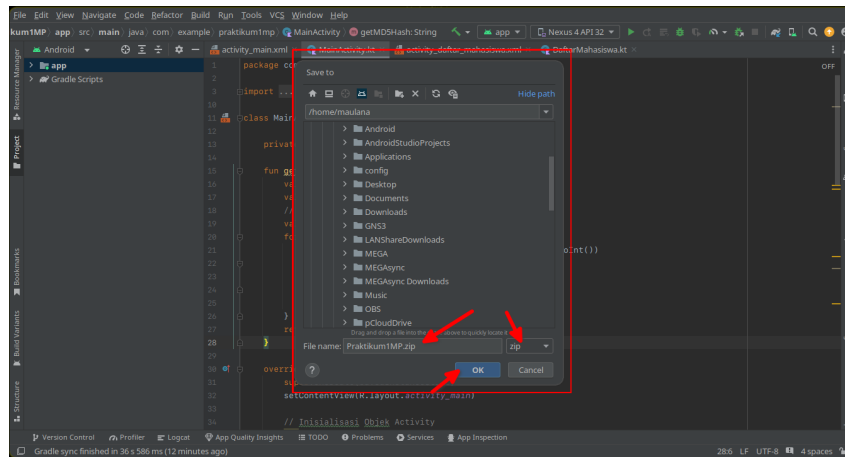
30. Perangkat siap digunakan dan ditampilkan



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



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



Bab 3

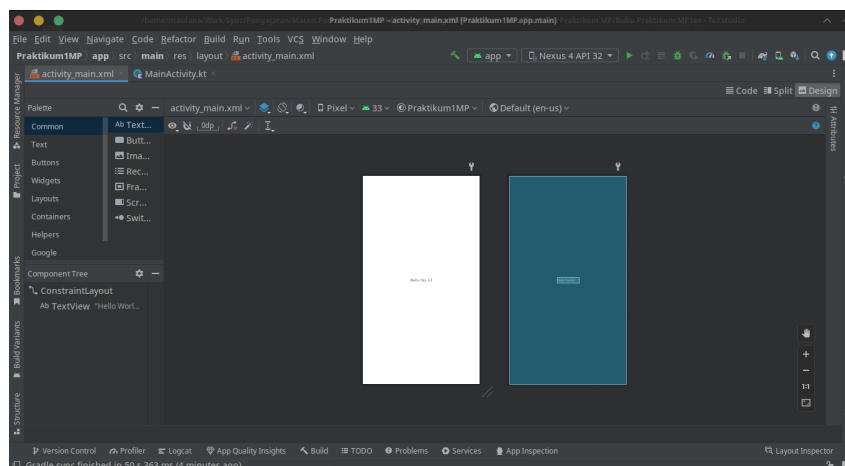
Praktikum 3

3.1 Antarmuka Halaman Login

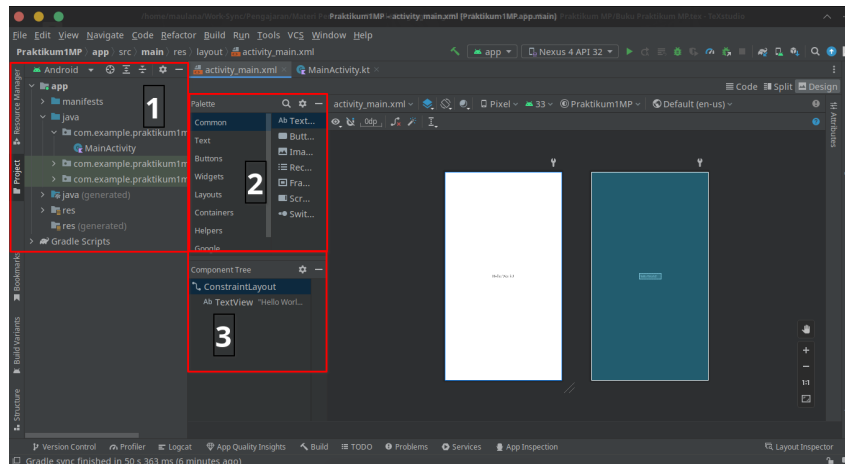
Di bagian ini mahasiswa akan diperkenalkan dengan antarmuka **Android Studio**, sehingga dapat membantu siswa dalam melakukan navigasi yang ada di dalam aplikasi tersebut.

3.2 Tutorial

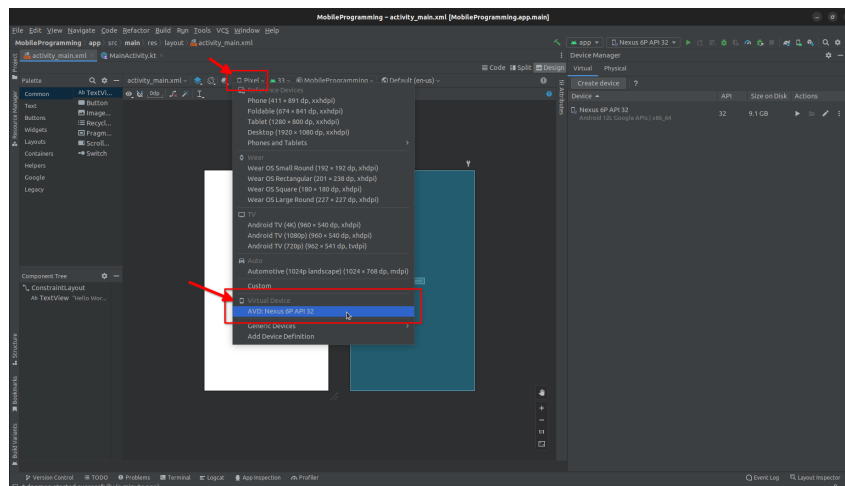
1. Buka **Android Studio**, dan buka kembali **Praktikum1MP** yang sudah dibuat sesi sebelumnya.



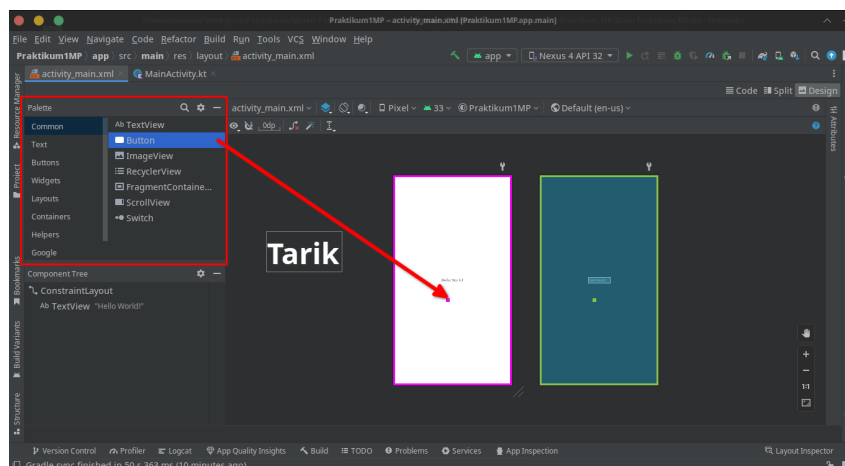
2. Di bagian panel kiri ada dua (2) hal yang tidak boleh dilupakan:
 - (a) Panel **Project** : Menampilkan projek saat ini (bisa dikecilkan)
 - (b) Panel **Palette** : Menampilkan objek yang bisa ditempel
 - (c) Panel **Component Tree** : Menampilkan objek yang sudah ditempel



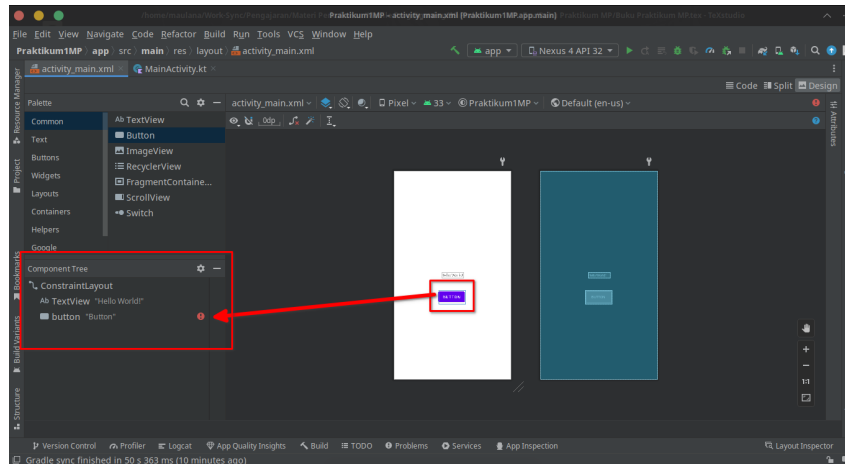
3. Agar **Activity** memiliki ukuran yang sesuai dengan ukuran perangkat, dapat difigurasi melalui tulisan **Pixel** di atas **Activity**, lalu pilih **Virtual Device: AVD**. Activity akan merubah ukurannya secara otomatis



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



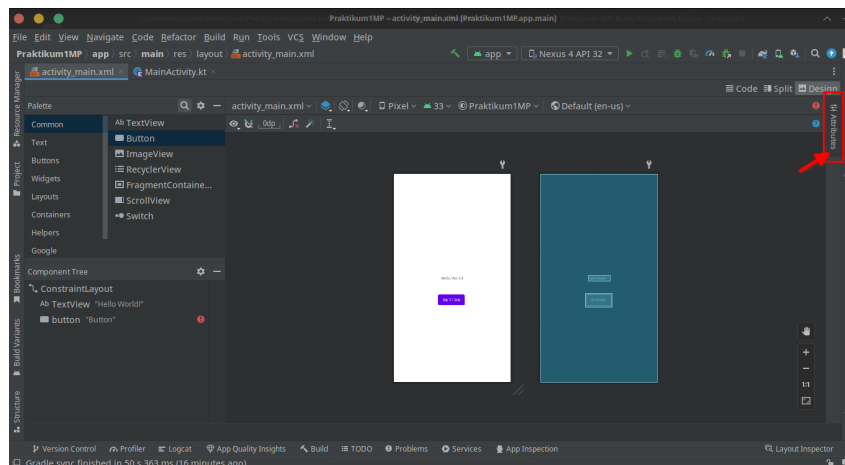
5. Jika dilakukan dengan benar, maka **Tombol** akan terlihat di **Kanvas Putih** dan **Component Tree** di Panel Kiri

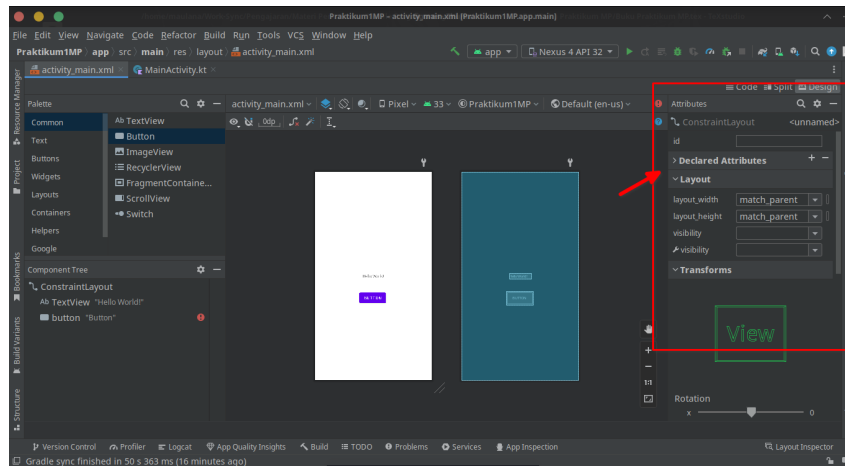


6. Namun objek ini masih memiliki **error** yang terlihat di **Component Tree**. Biasanya error ini muncul karena:

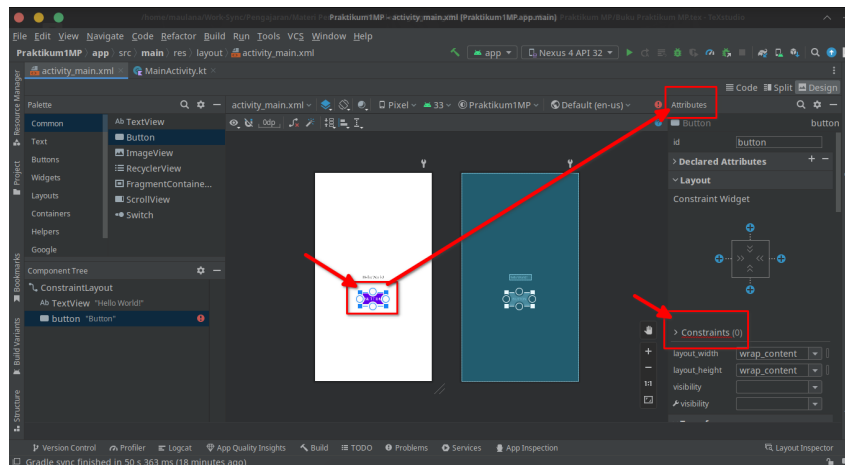
- (a) 1 **Constraint** Vertikal dan 1 **Contraint** Horizontal tidak diatur
- (b) Ukuran **Tombol** terlalu kecil

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

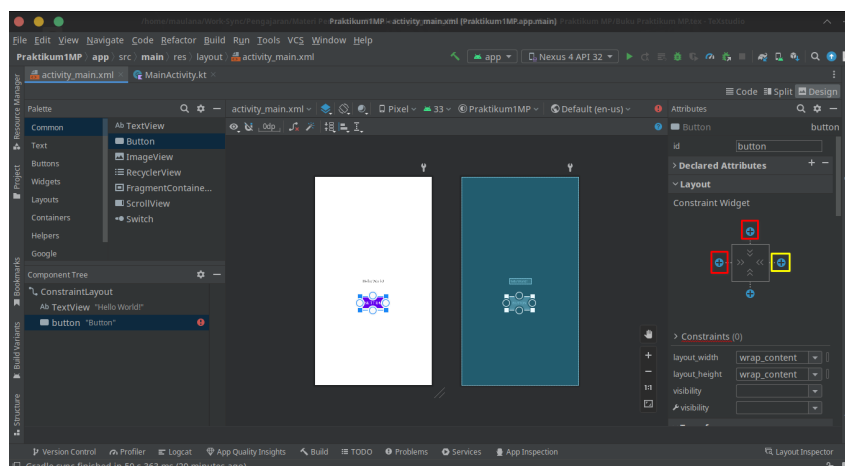




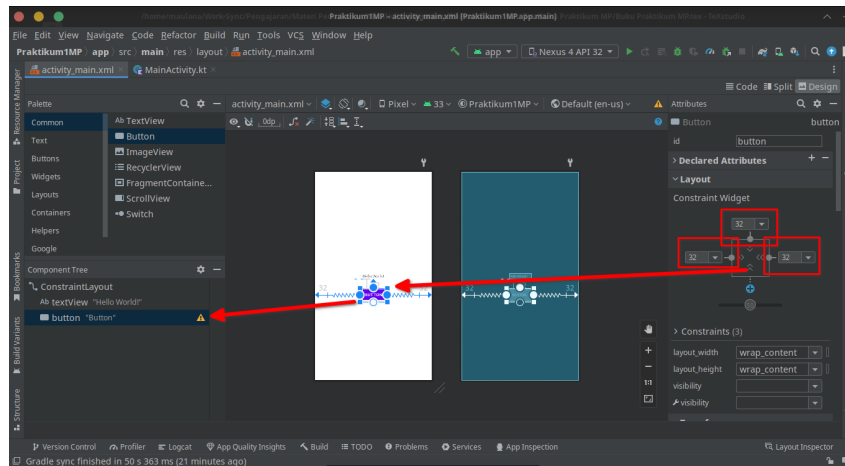
8. Untuk mengkonfigurasi **Attributes** dari suatu objek, pastikan panel **Attributes** terbuka, dan klik **Objek** yang mau dirubah. Berdasarkan gambar di bawah, **Constraint** tidak di set sehingga muncul indikator **Merah**



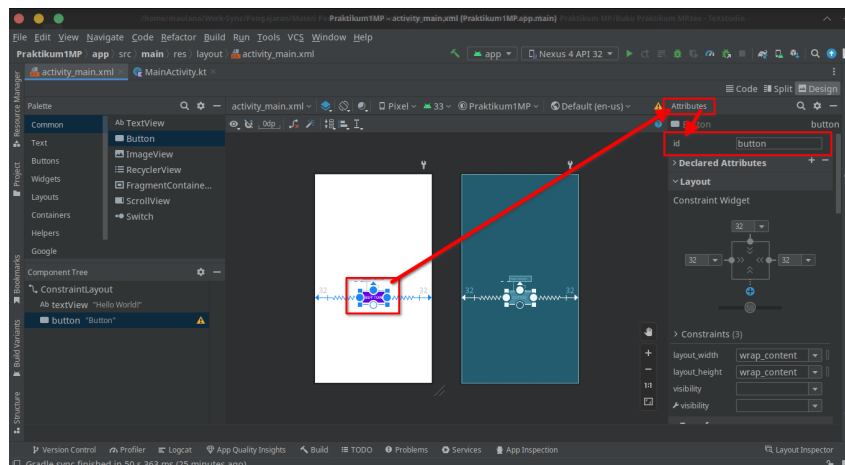
9. Untuk memperbaiki ini, cukup tambahkan **Constraint Atas** dan **Constraint Samping** seperti gambar berikut (Merah Wajib, Kuning Opsional):



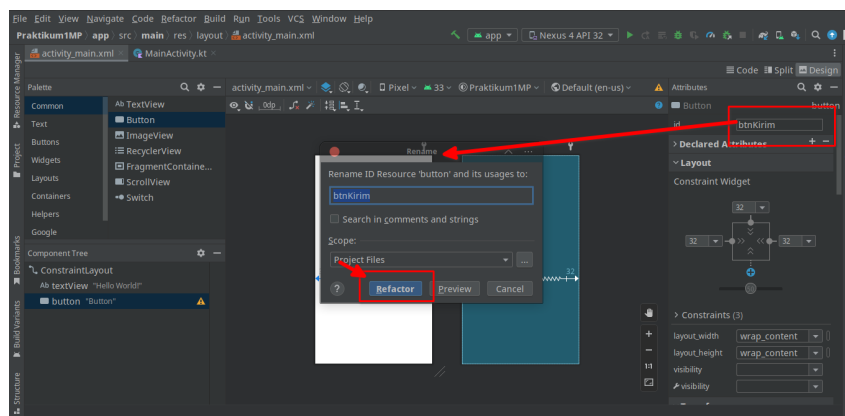
10. Atur jarak **Constraint** secara manual (Contoh 32) jika diperlukan, dan **error** pada objek **Tombol** sudah hilang.



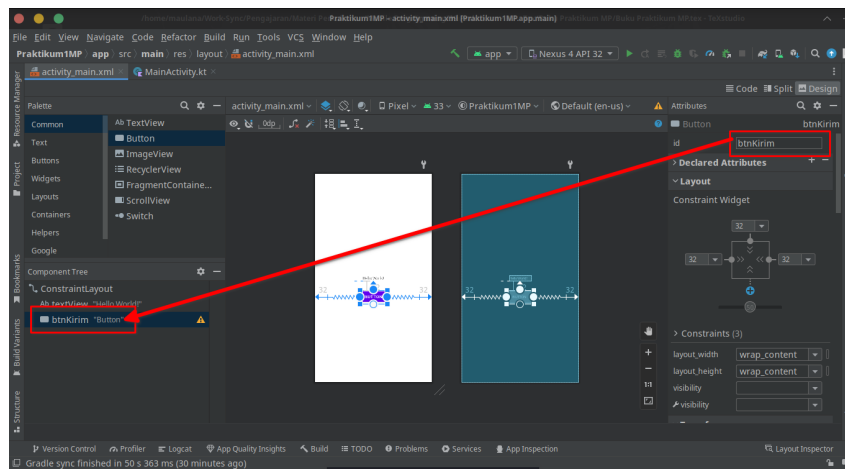
11. Dalam aplikasi Android memiliki banyak kemungkinan untuk ditambahkan objek dalam jumlah yang banyak. Maka untuk membedakan objek satu dengan yang lainnya dapat menggunakan **Attribute ID**
12. Untuk mengatur **ID Objek** cukup klik **Objek** dan lihat panel **Attribute**. Naikkan panel sampai paling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



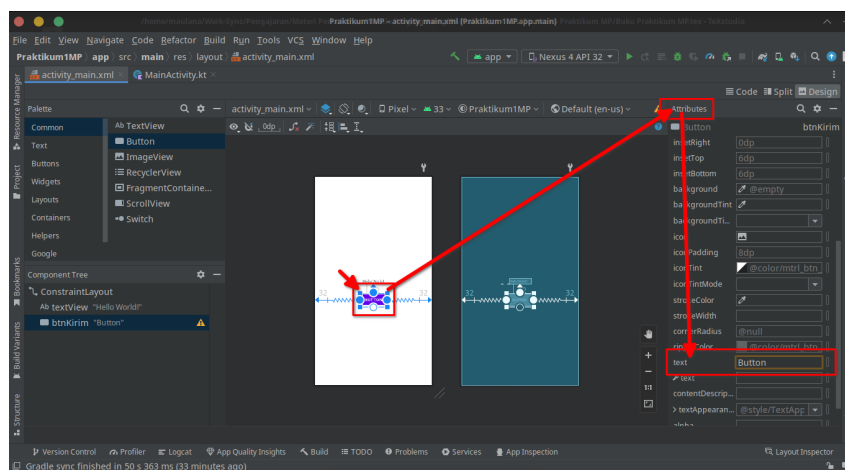
13. Ubah **ID Objek** menjadi **btnKirim** (atau nama unik lainnya). Klik area bebas maka permintaan **Rename** akan muncul. Klik **Refactor** untuk menerapkan penamaan ID Objek.



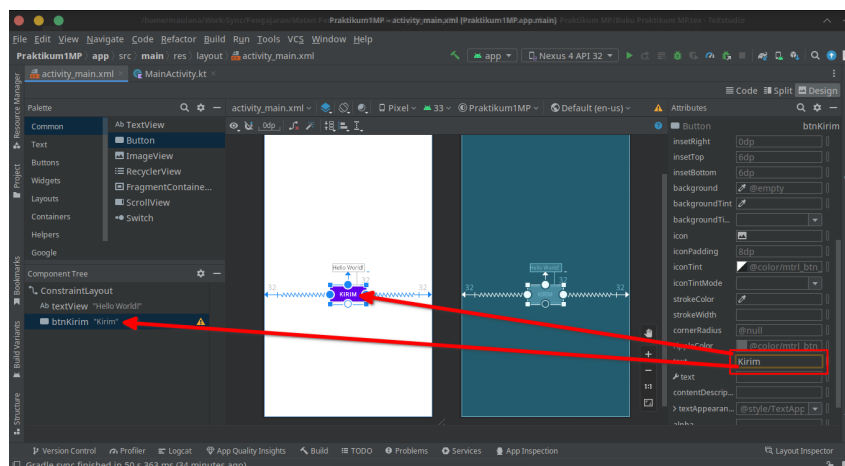
14. Penamaan ini akan langsung berdampak ke **Component Tree**. Maka untuk mengakses **Tombol** ini cukup memanggil **ID Objek** tersebut



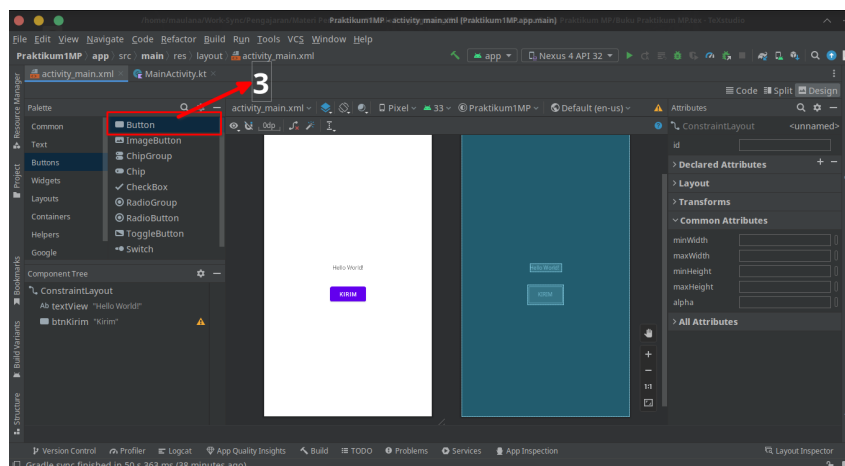
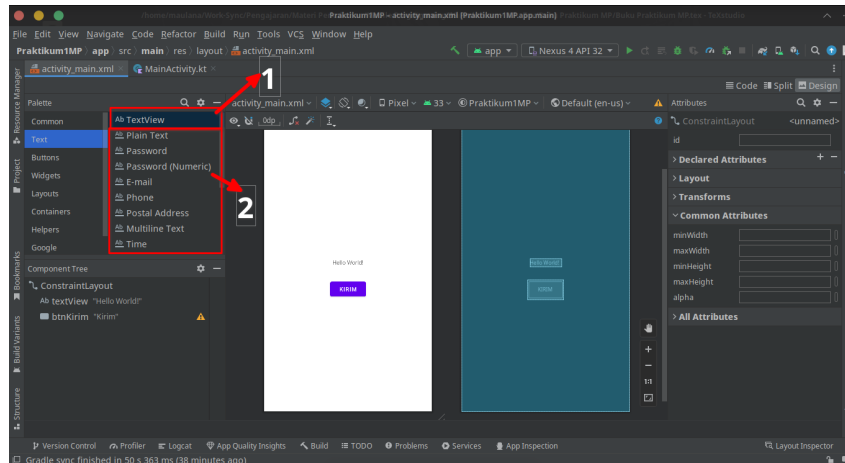
15. Berikutnya adalah mengubah **Isi Teks**. Dengan cara yang sama namun ketika memilih panel **Attribute**, pilih **Text** (Gunakan **Hint** untuk **EditText**)



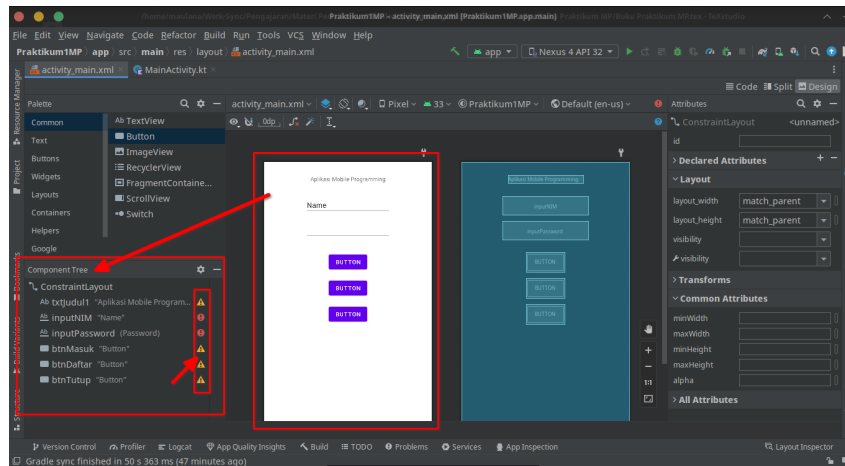
16. Isi langsung dengan teks yang diinginkan lalu **Enter**



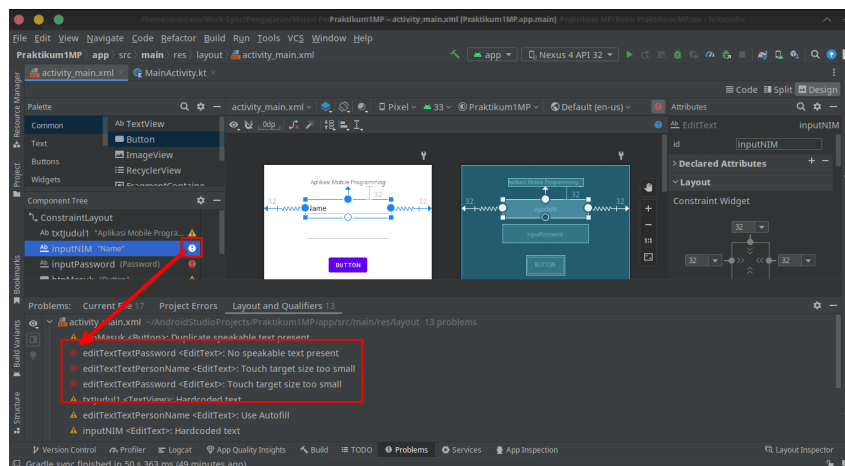
17. Konfigurasi ini berlaku untuk **Semua Objek**. Sehingga memudahkan dalam konfigurasinya.
18. Selain itu ada beberapa objek lain yang wajib diketahui berupa:
- (a) **TextView** : Digunakan untuk menampilkan teks / String saja tanpa input
 - (b) **EditText** : Digunakan untuk menampilkan dan menerima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

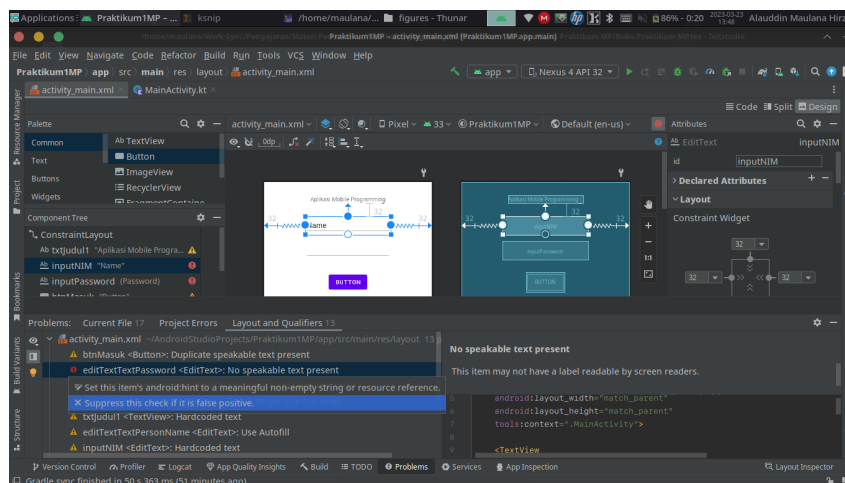


20. Ketika semua terpasang, maka akan muncul khususnya untuk **EditText** nya. Klik **Indikator Error Merah** untuk menampilkan **error**.

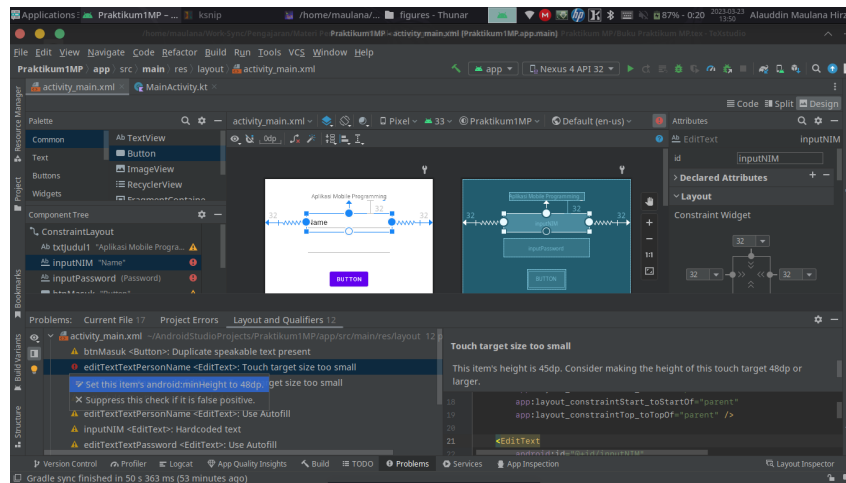


21. Untuk memperbaiki ini, gunakan **Klik Kanan**, pilih **Show Quick Fixes** lalu pilih:

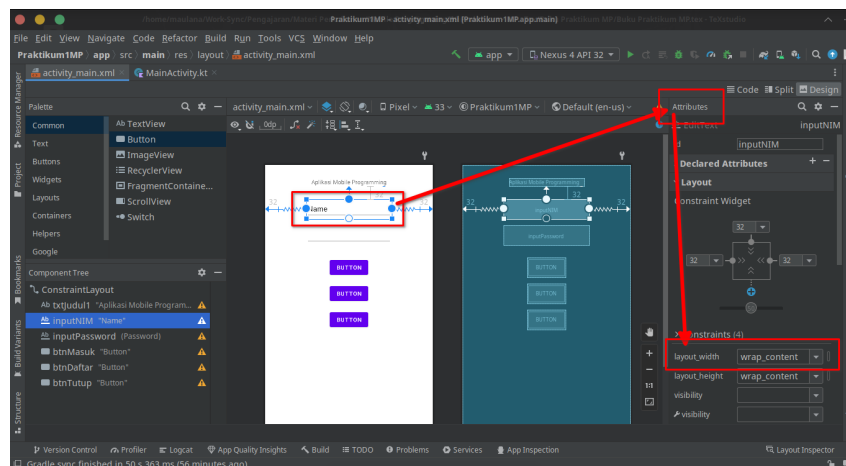
- Jika **Error No speakable text present** → **Suppress this check if it is false positive**



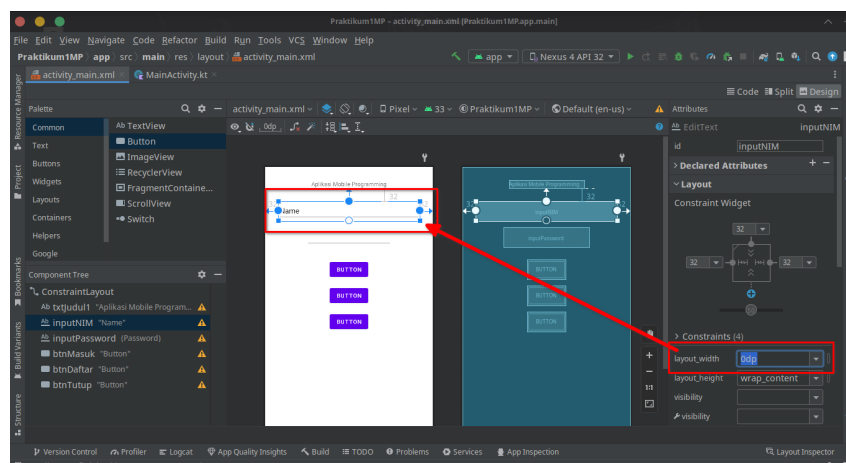
- Jika **Error Touch target size too small** → Set this item's android:minHeight to 48dp



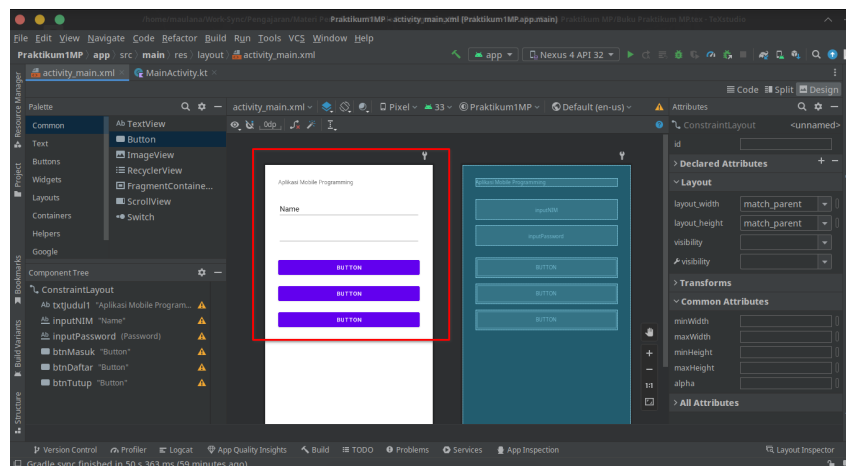
22. Error akan hilang. Namun tampilan terlalu jelek. Untuk mengatur **Lebar Objek** cukup klik **Object** → **Attribute** → **layout_width**



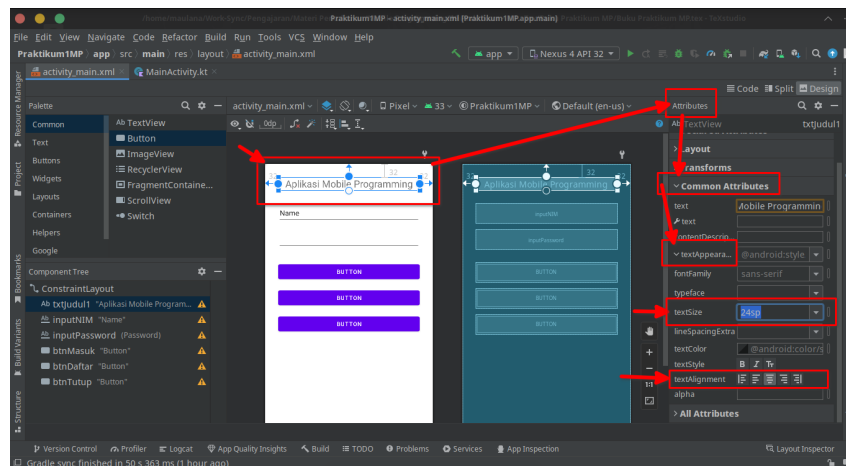
23. Ubah dari **wrap_content** ke **0dp** atau **match_constraint** untuk melebarkan objek se lebar **constraint**



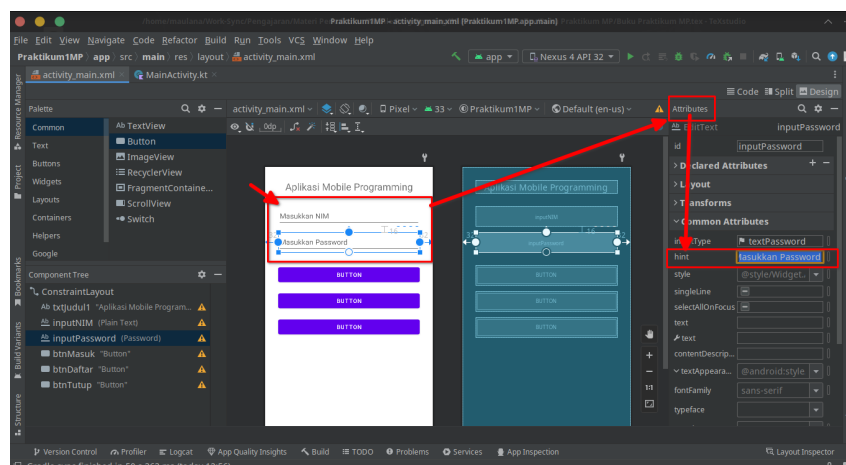
24. Ulangi ke **Semua Objek** hingga terlihat seperti berikut



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

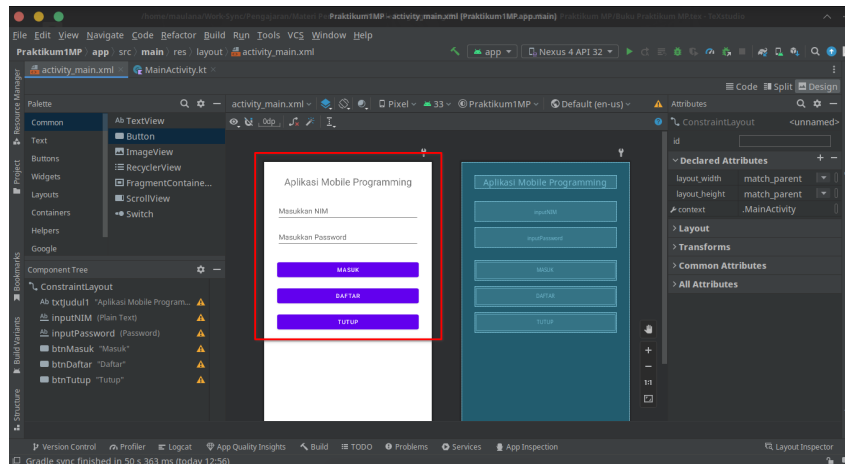


26. Ubah **Hint EditText** (inputNIM) dan **EditText** (inputPassword) menjadi seperti berikut (Pastikan **Attribute Text** Kosong):



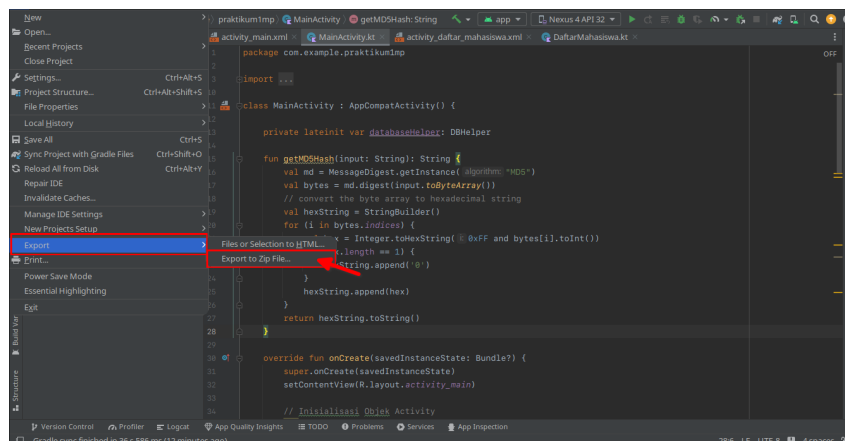
27. Terakhir, ubah **Teks** dari masing-masing **Tombol** menjadi:

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

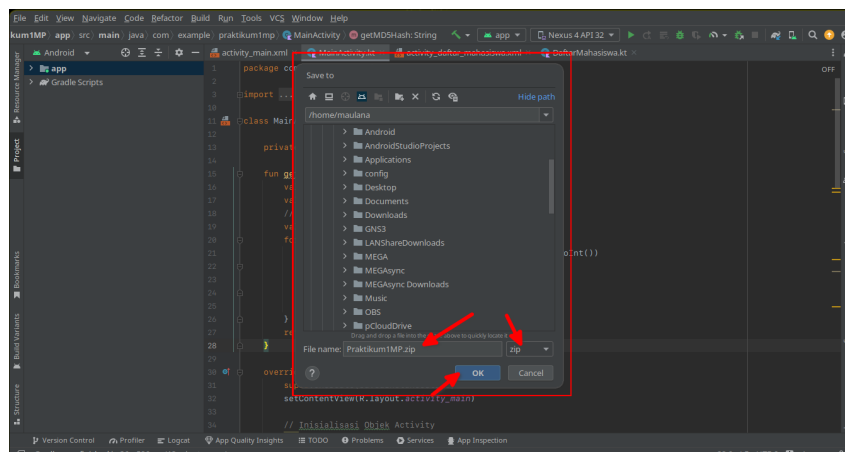


28. Screenshot hasil dan kirimkan ke **E-Learning**

29. Untuk mengirimkan hasilnya, 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 4

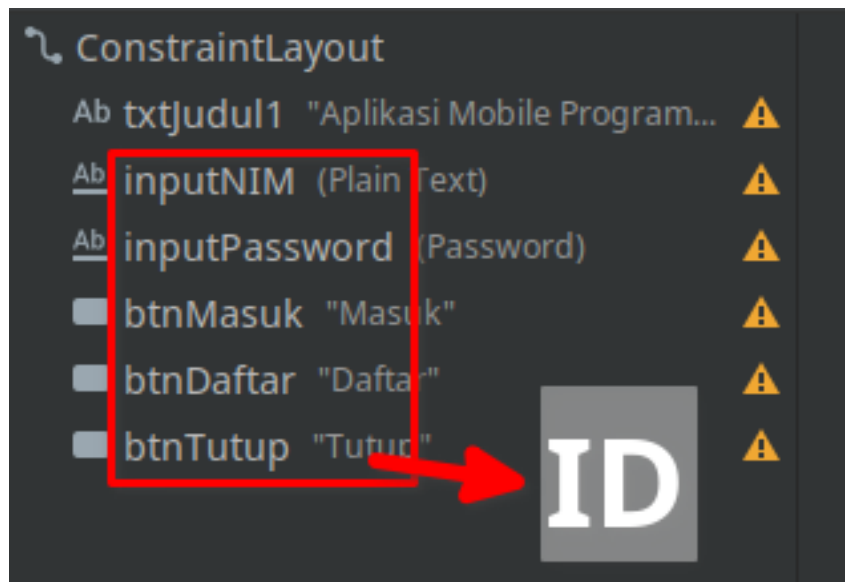
Praktikum 4

4.1 Pemrograman Halaman Login

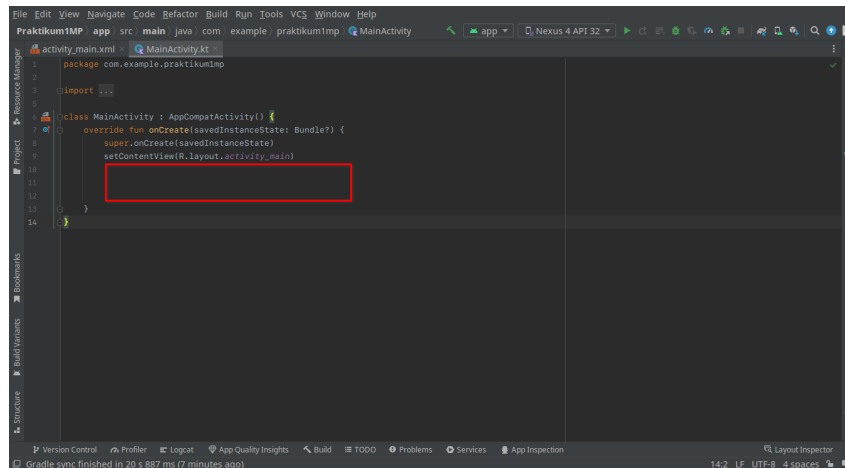
Di bagian ini mahasiswa akan diperkenalkan dengan antarmuka **Android Studio**, sehingga dapat membantu siswa dalam melakukan navigasi yang ada di dalam aplikasi tersebut.

4.2 Tutorial

1. Untuk memulai memprogram objek-objek yang ada di dalam Activity, buka lah proyek 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



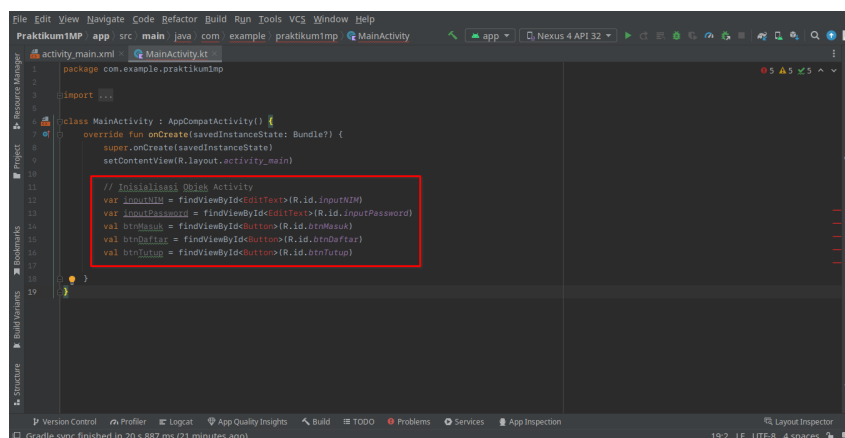
4. Masukkan kode berikut untuk menginisialisasi / mengubah objek menjadi variabel kode

Potongan Kode

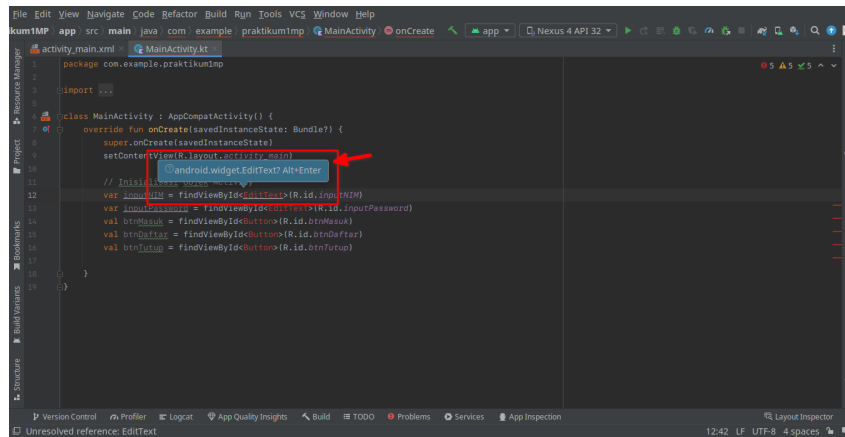
```

// 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)

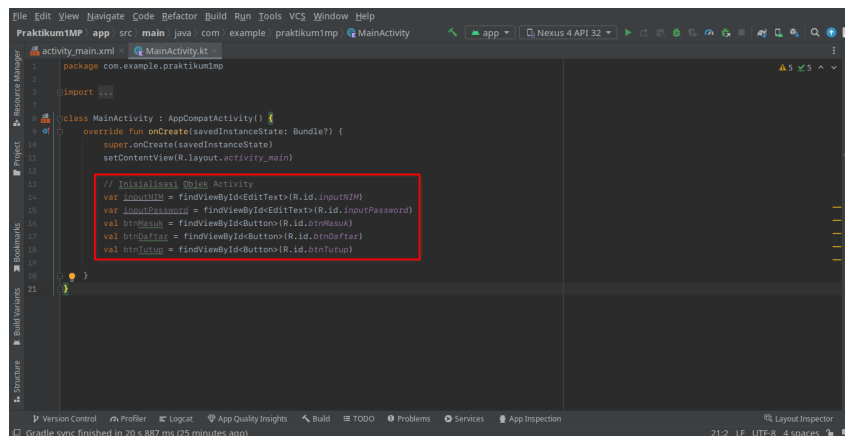
```



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 kode adalah disengaja

Potongan Kode

```

btnMasuk.setOnClickListener {

}

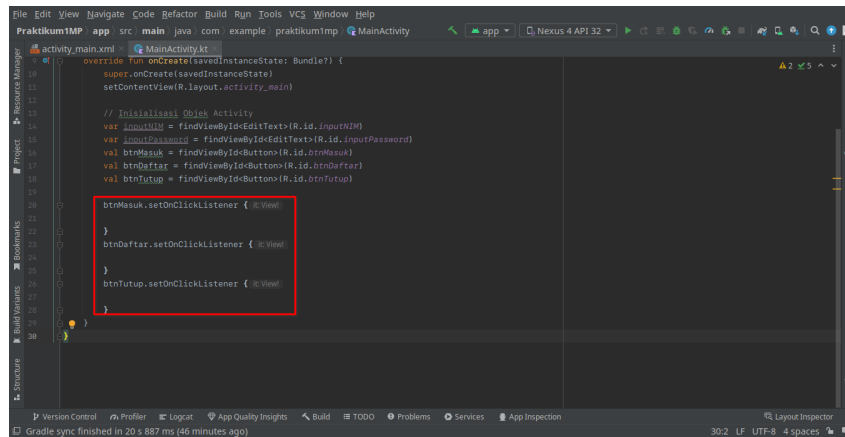
btnDaftar.setOnClickListener {

}

btnTutup.setOnClickListener {

}

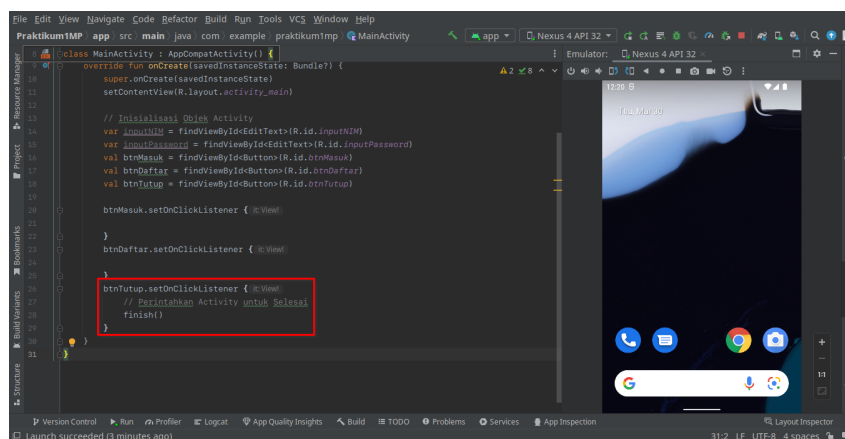
```



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

Potongan Kode

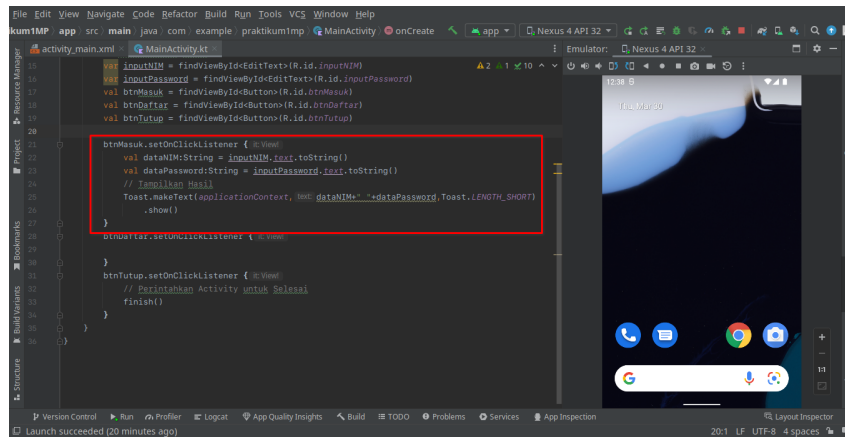
```
// Perintahkan Activity untuk Selesai
finish()
```



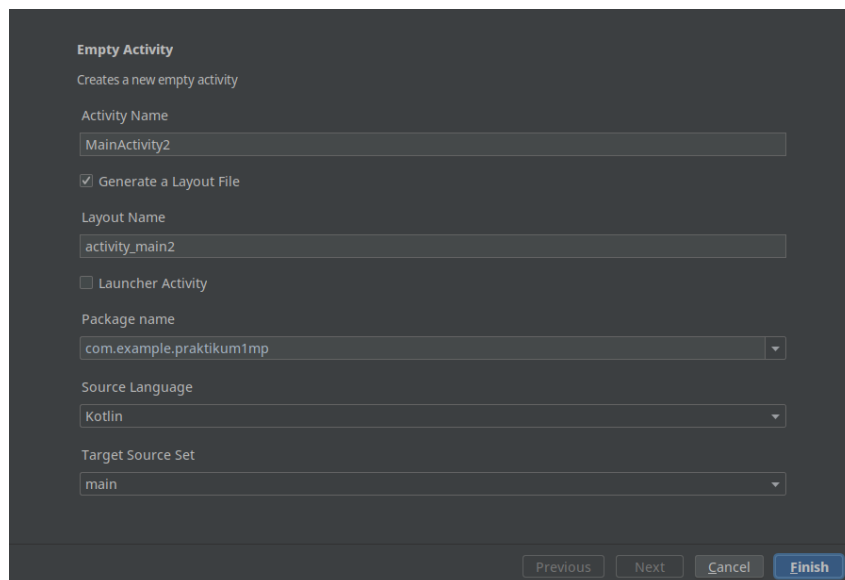
9. Kode berikutnya adalah untuk mengambil data login dari **EditText** yang ditempelkan di **activity_main.xml**. Namun saat ini kode hanya bisa menampilkan saja.

Potongan Kode

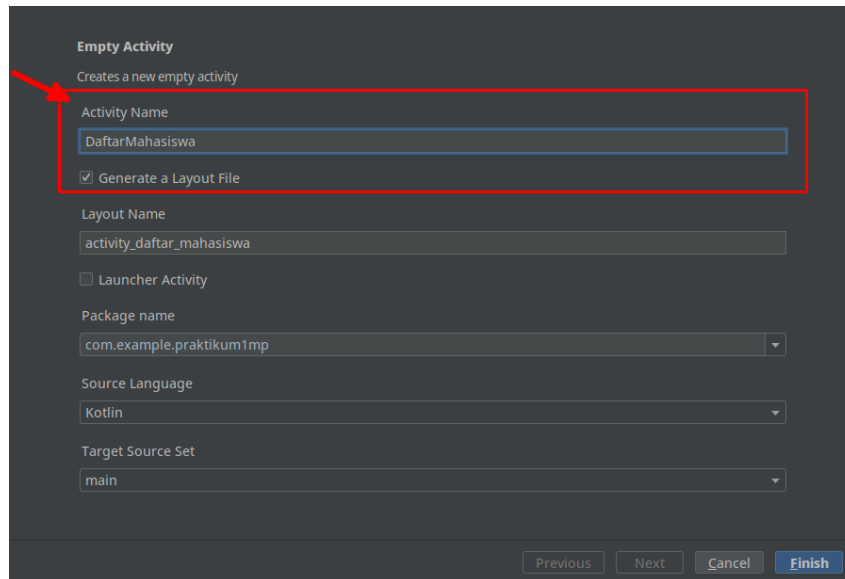
```
val dataNIM:String = inputNIM.text.toString()
val dataPassword:String = inputPassword.text.toString()
// Tampilkan Hasil
Toast.makeText(applicationContext,dataNIM+" "+dataPassword,Toast.LENGTH_SHORT).show()
```



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



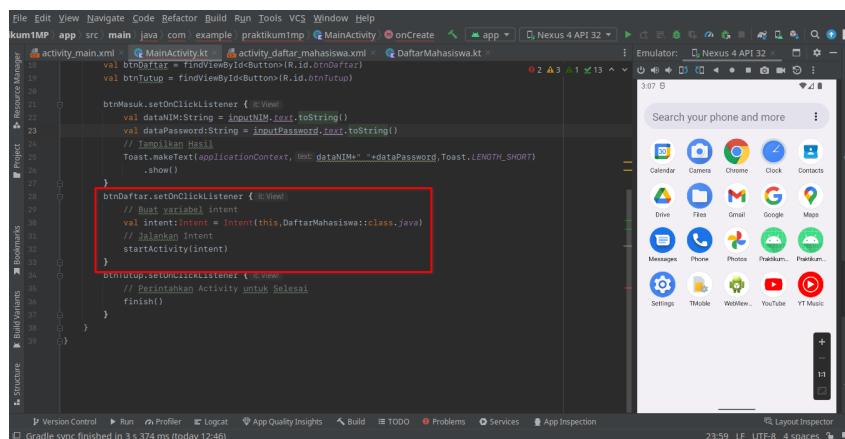
11. Ubah **MainActivity2** menjadi **DaftarMahasiswa** di bagian **Activity Name**. Sisanya akan mengikuti. Lalu klik **Finish**



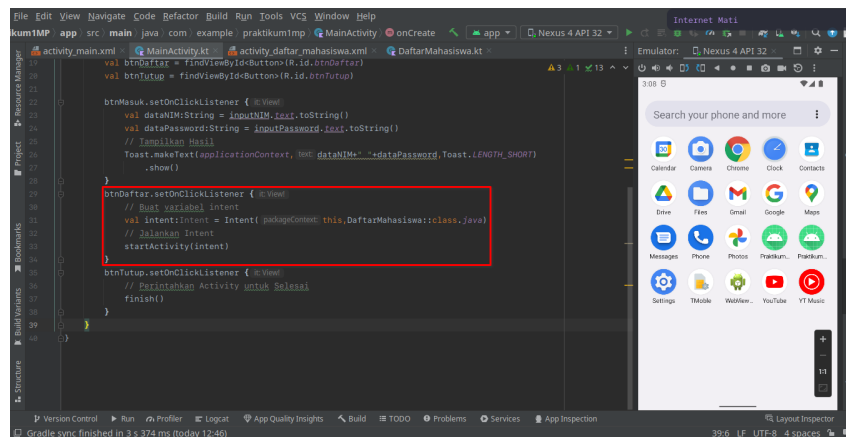
12. Android Studio akan menampilkan layout dan kode kotlin **Activity** baru ini. Kembali buka **MainActivity.kt**
13. Masukkan kode baru untuk tombol **btnDaftar** dengan kode seperti berikut:

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**

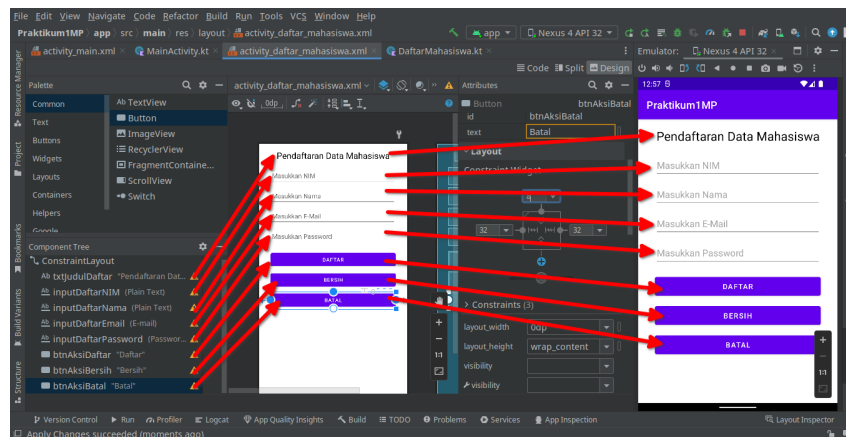


15. Tes aplikasi dengan menjalankannya ke **Emulator**
16. Jika sudah berhasil, tambahkan objek-objek ke layout **activity_daftar_mahasiswa.xml** seperti berikut:

- **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

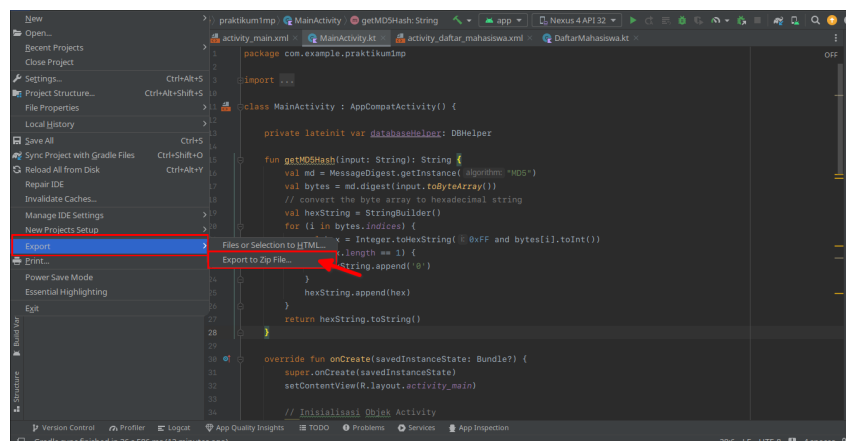
- **Button**

- ID : btnAksiBatal
- Text : Batal

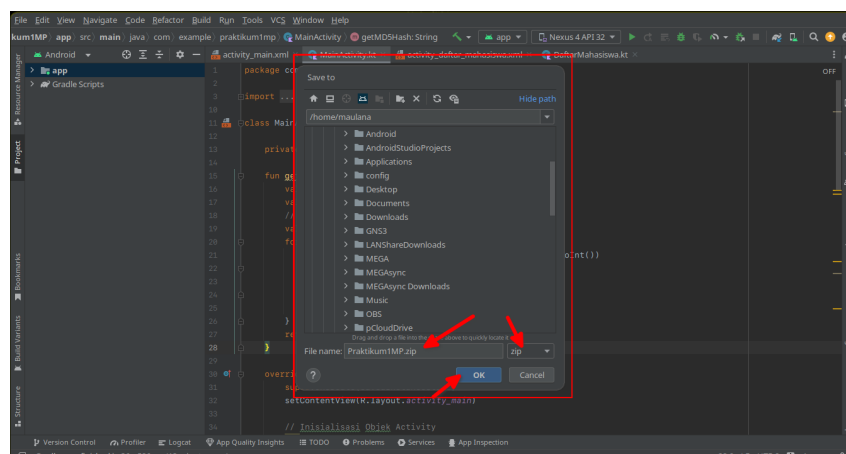


17. Jalan kan aplikasi, dan untuk berpindah ke halaman berikutnya klik tombol **Daftar**
Screenshot hasil dan kirimkan ke **E-Learning**

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 5

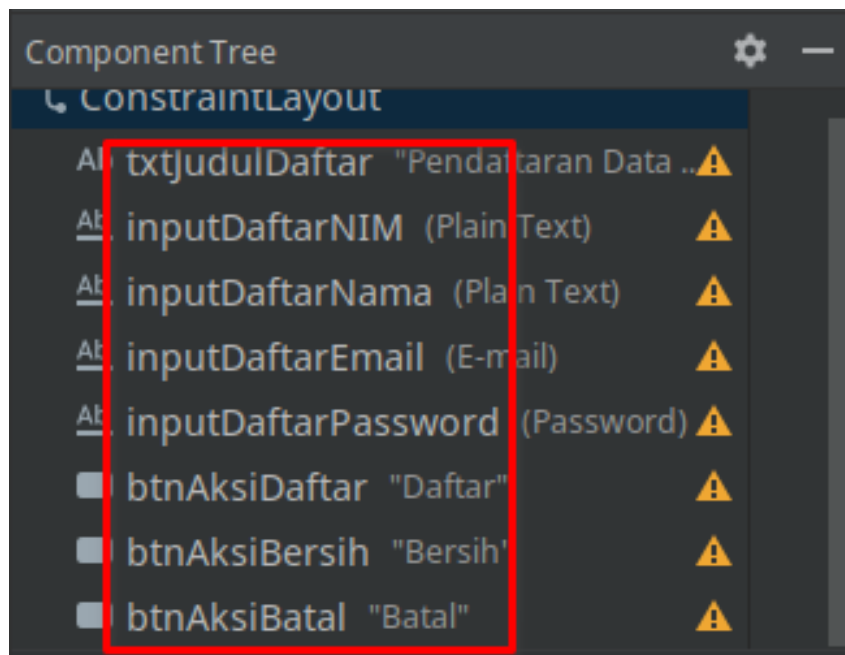
Praktikum 5

5.1 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.

5.2 Tutorial

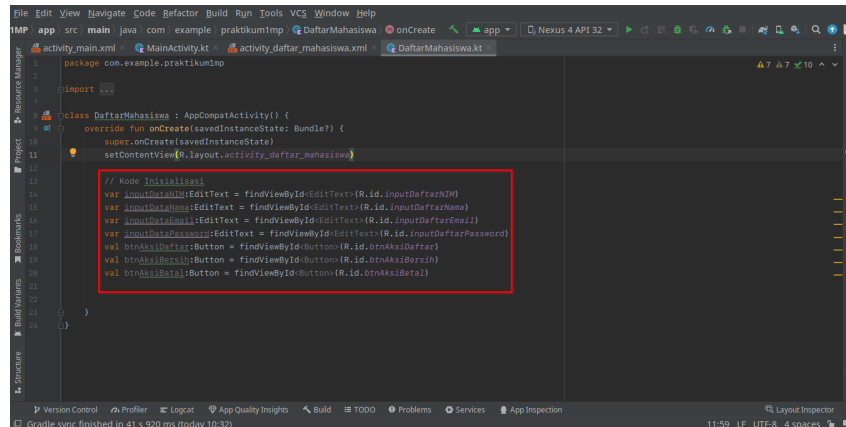
1. Buka proyek sebelumnya yang sudah memiliki **Activity** Pendaftaran
2. Pastikan **ID** yang ada di dalam file **activity_daftar_mahasiswa.xml** persis dengan gambar berikut:



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

Potongan Kode

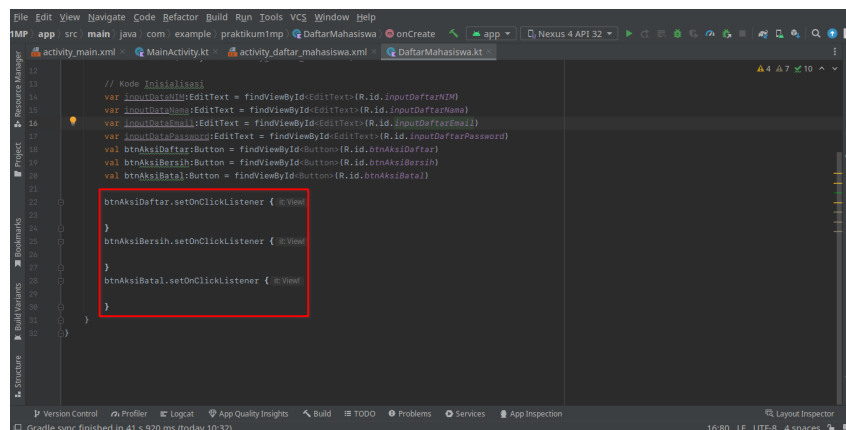
```
// Kode Inisialisasi
var inputDataNIM:EditText = findViewById<EditText>(R.id.inputDaftarNIM)
var inputDataNama:EditText = findViewById<EditText>(R.id.inputDaftarNama)
var inputDataEmail:EditText = findViewById<EditText>(R.id.inputDaftarEmail)
var inputDataPassword:EditText = findViewById<EditText>(R.id.inputDaftarPassword)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



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

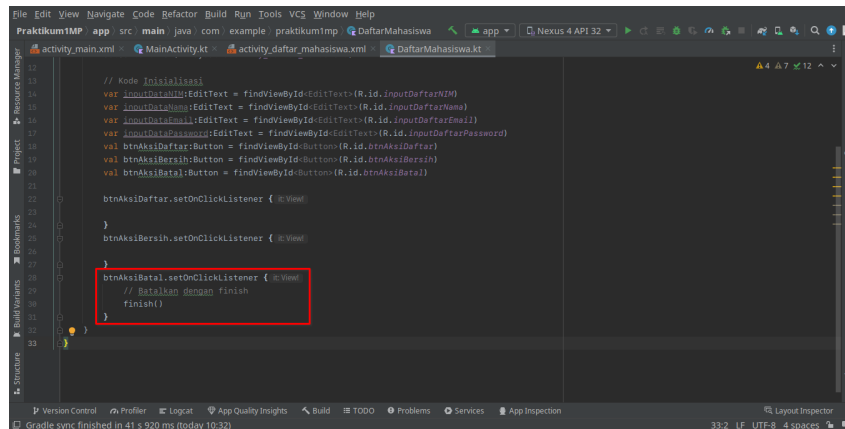
```
btnAksiDaftar.setOnClickListener {
}
btnAksiBersih.setOnClickListener {
}
btnAksiBatal.setOnClickListener {
}
```



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

finish()

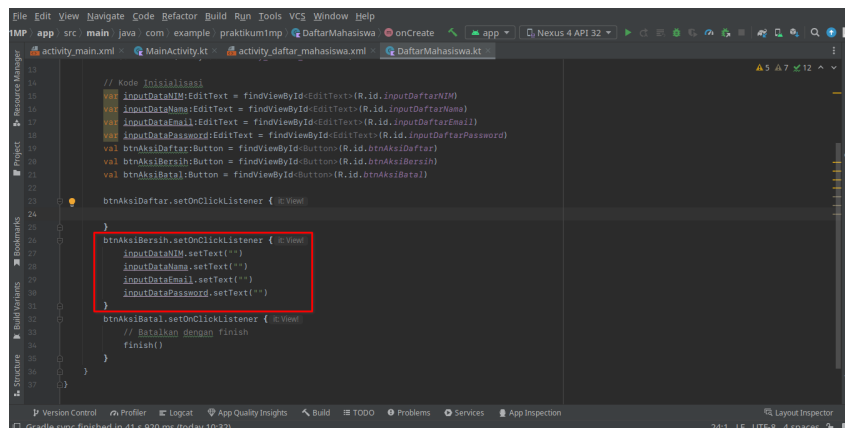
Potongan Kode



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

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

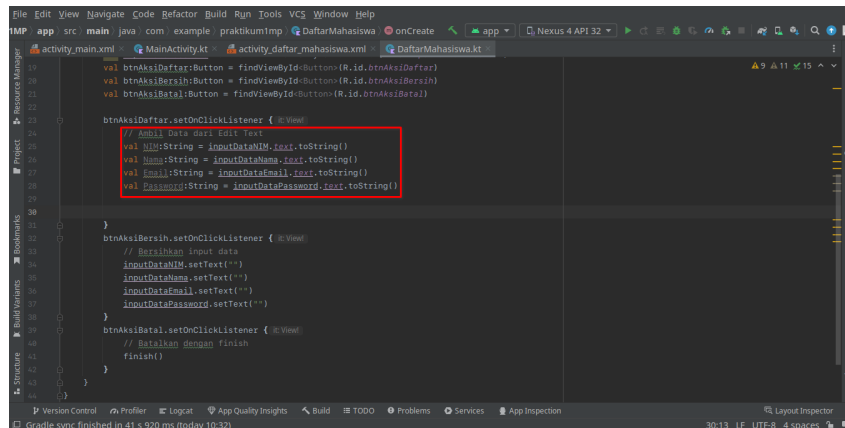
Potongan Kode



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



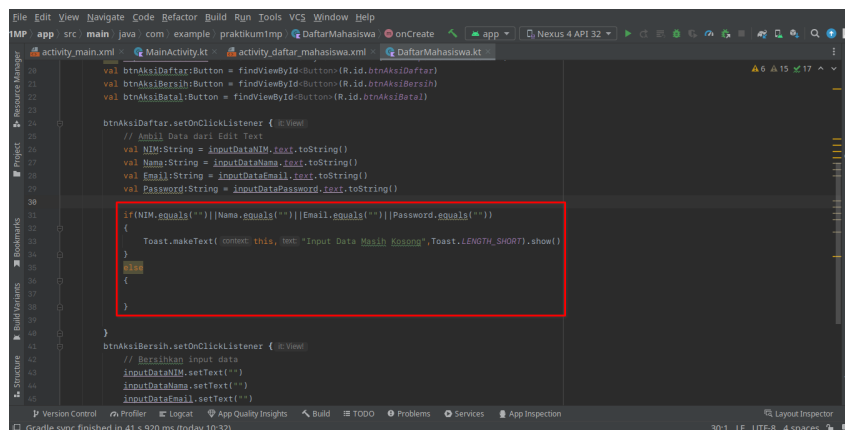
9. Jika sudah memasukkan kode untuk mengambil data input, maka tahap berikutnya adalah melakukan validasi input tadi dengan kode berikut:

Potongan Kode

```

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

```



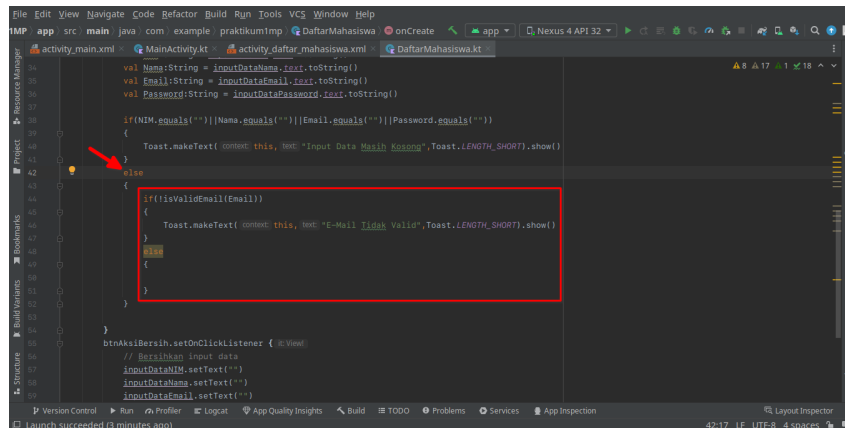
10. Sesudah memberikan logika pengecekan input kosong, berikutnya adalah melakukan validasi E-Mail.
11. Tambahkan **import** untuk pengecekan E-Mail

Potongan Kode

```

import android.util.Patterns

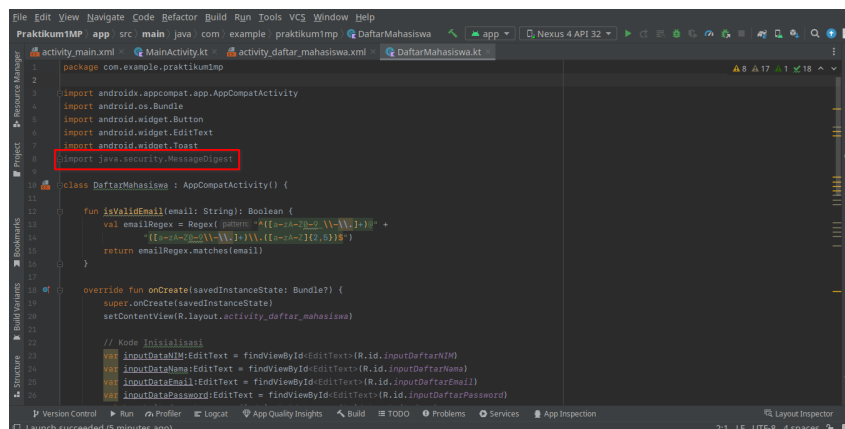
```

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

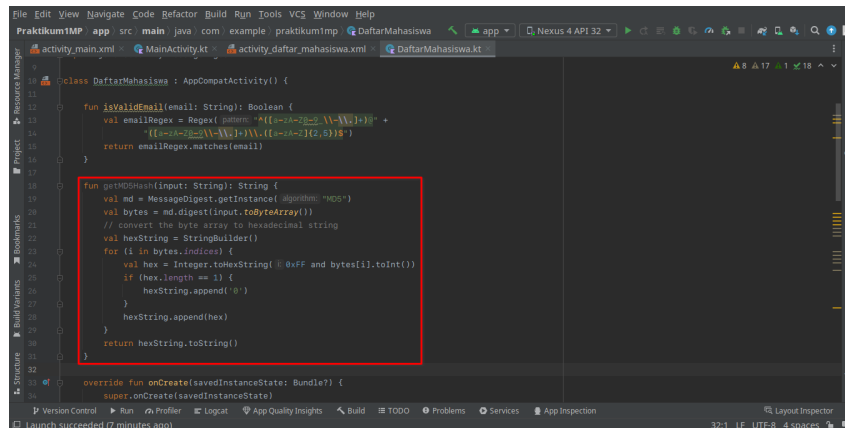
```
import java.security.MessageDigest
```



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()
}
```



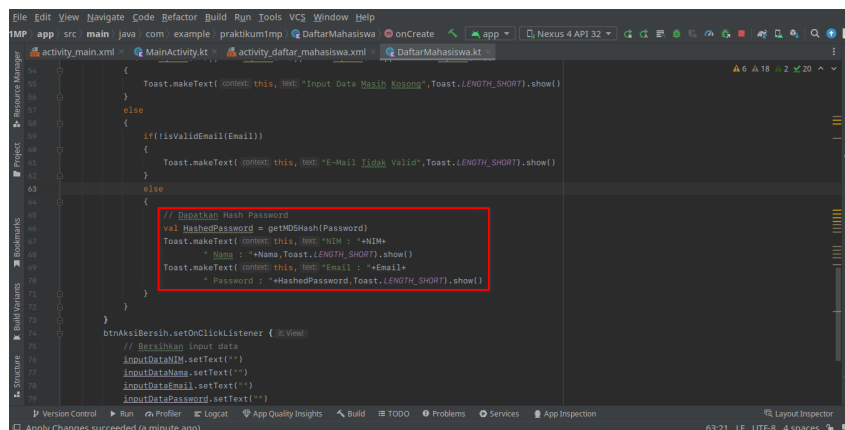
16. Kode ini kemudian diakses melalui kode yang ditambahkan di dalam **ELSE** dari kode langkah 11. Perhatikan kode dan gambar berikut:

Potongan Kode

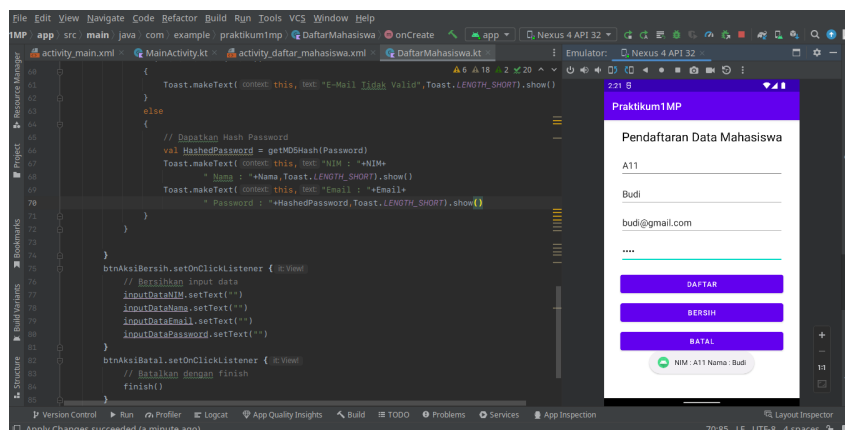
```

// 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()

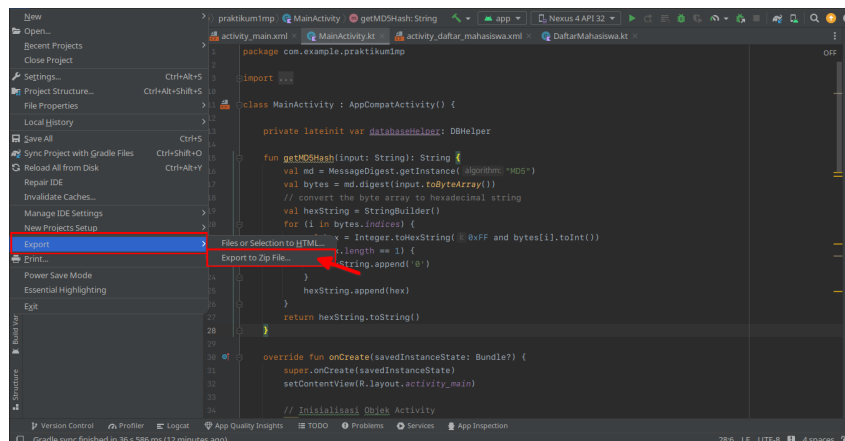
```



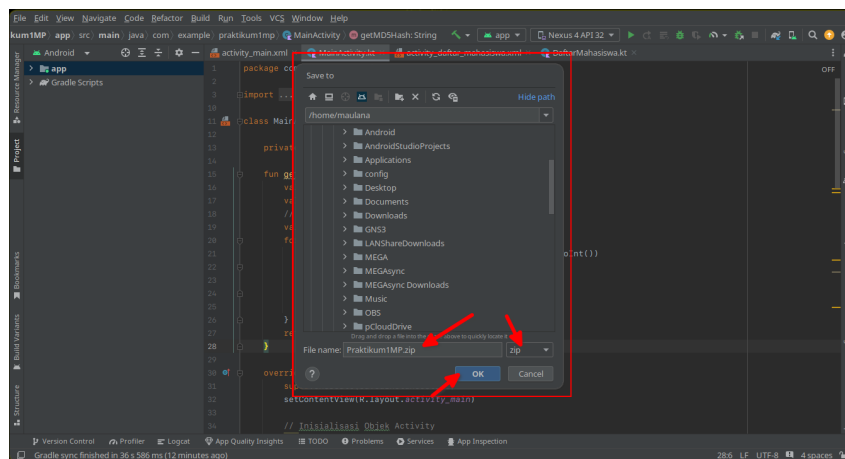
17. Uji coba aplikasi dengan memasukkan data di bagian Pendaftaran



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 6

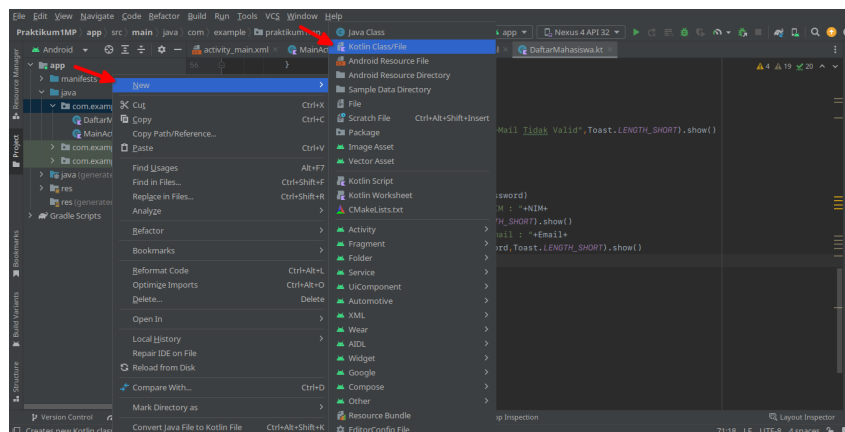
Praktikum 6

6.1 SQLite Insert Data

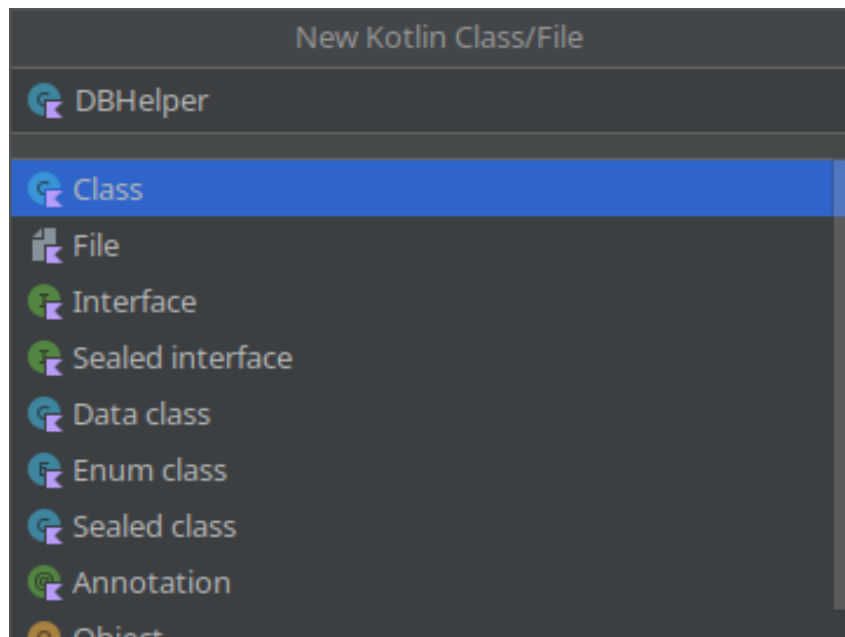
Di bagian ini mahasiswa diajarkan bagaimana melakukan input data melalui Form yang telah dibuat dan mengirimkannya ke database internal SQLite. Mahasiswa diwajibkan menyelesaikan Praktikum 4.

6.2 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,
        // Untuk upgrade database
        newVersion: Int) {
    }

}
```

```

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, 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 }

```

5. Buka file **DaftarMahasiswa.kt**

```

57     else
58     {
59         if (isValidEmail(email))
60         {
61             Toast.makeText(this, 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(this, R.string.HASH : "+NIM+",
68                 "Nama : "+name, Toast.LENGTH_SHORT).show()
69             Toast.makeText(this, R.string.Email : "+Email+",
70                 "Password : "+hashedPassword, Toast.LENGTH_SHORT).show()
71         }
72     }
73
74     btnAksiBersih.setOnClickListener { //View
75         // Bersihkan input data
76         inputDataNIM.setText("")
77         inputDataEmail.setText("")
78         inputDataPassword.setText("")
79     }
80
81     btnAksiBatal.setOnClickListener { //View
82         // Batalkan proses login
83     }
84 }

```

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
3  import androidx.appcompat.app.AppCompatActivity
4
5  class DaftarMahasiswa : AppCompatActivity() {
6
7      private lateinit var databaseHelper: DBHelper
8
9      fun isValidEmail(email: String): Boolean {
10         val result: Boolean = Patterns.EMAIL_ADDRESS.matcher(email).matches()
11         return result
12     }
13
14     fun getMD5Hash(input: String): String {
15         val md = MessageDigest.getInstance("MD5")
16         val bytes = md.digest(input.toByteArray())
17         // convert the byte array to hexadecimal string
18         val hexString = StringBuilder()
19         for (i in bytes.indices) {
20             val hex = Integer.toHexString(0xFF and bytes[i].toInt())
21             if (hex.length == 1) {
22                 hexString.append("0")
23             }
24             hexString.append(hex)
25         }
26         return hexString.toString()
27     }
28 }

```

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

Potongan Kode
databaseHelper = DBHelper(this)

```

36  override fun onCreate(savedInstanceState: Bundle?) {
37      super.onCreate(savedInstanceState)
38      setContentView(R.layout.activity_daftar_mahasiswa)
39
40      // Kode Inisialisasi
41      val inputDaftarNM: EditText = findViewById<EditText>(R.id.inputDaftarNM)
42      val inputDaftarNama: EditText = findViewById<EditText>(R.id.inputDaftarNama)
43      val inputDaftarEmail: EditText = findViewById<EditText>(R.id.inputDaftarEmail)
44      val inputDaftarPassword: 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      databaseHelper = DBHelper(context);
50
51      btnAksiDaftar.setOnClickListener { @View
52          // Ambil Data dari Edit Text
53          val NM:String = inputDaftarNM.text.toString()
54          val Nama:String = inputDaftarNama.text.toString()
55          val Email:String = inputDaftarEmail.text.toString()
56          val Password:String = inputDaftarPassword.text.toString()
57
58          if(NM.equals("") || Nama.equals("") || Email.equals("") || Password.equals(""))
59          {
60              Toast.makeText(context, "Input Data Masih Kosong", Toast.LENGTH_SHORT).show()
61          }
62      }
63  }

```

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

```

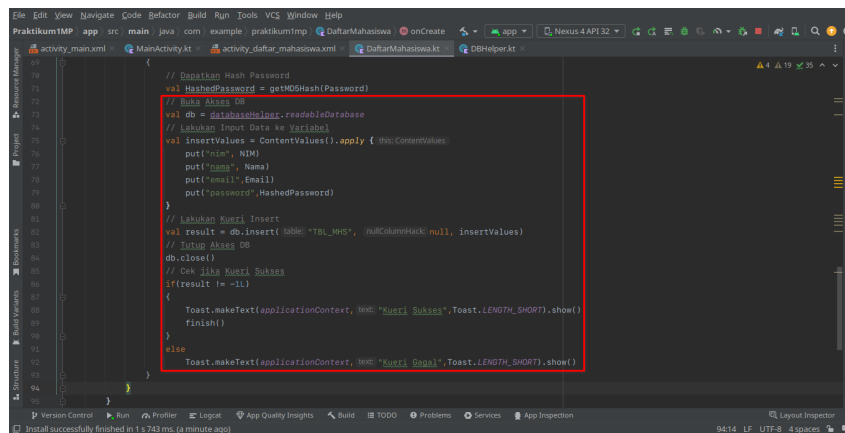
58  {
59      Toast.makeText(context, "Input Data Masih Kosong", Toast.LENGTH_SHORT).show()
60  }
61  else
62  {
63      if(!isValidEmail(Email))
64      {
65          Toast.makeText(context, "E-Mail Tidak Valid", Toast.LENGTH_SHORT).show()
66      }
67      else
68      {
69          // Buat Hash Password
70          val HashedPassword = getMD5Hash(Password)
71          Toast.makeText(context, "NM : "+NM,
72              "Nama : "+Nama, Toast.LENGTH_SHORT).show()
73          Toast.makeText(context, "Email : "+Email,
74              "Password : "+HashedPassword, Toast.LENGTH_SHORT).show()
75      }
76  }
77  }
78
79  btnAksiBersih.setOnClickListener { @View
80      // Bersihkan input data
81      inputDaftarNM.setText("")
82      inputDaftarNama.setText("")
83      inputDaftarEmail.setText("")
84      inputDaftarPassword.setText("")
85  }
86  }

```

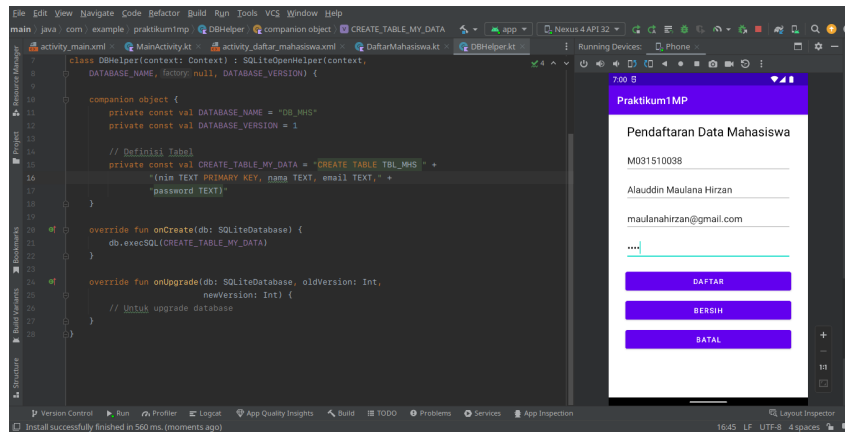
9. Tepat di bawah variabel **HashedPassword**, masukkan kode berikut:

Potongan Kode

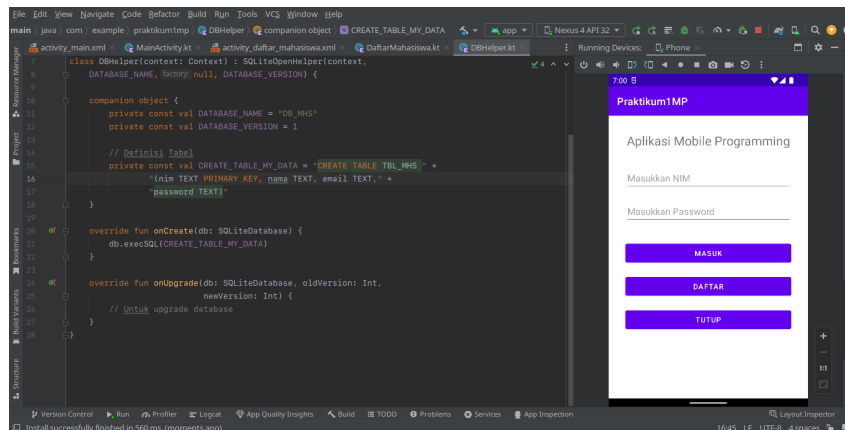
```
// Buka Akses DB
val db = databaseHelper.readableDatabase
// Lakukan Input Data ke Variabel
val insertValues = ContentValues().apply {
    put("nim", NIM)
    put("nama", Nama)
    put("email", Email)
    put("password", HashedPassword)
}
// Lakukan Kueri Insert
val result = db.insert("TBL_MHS", 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()
```



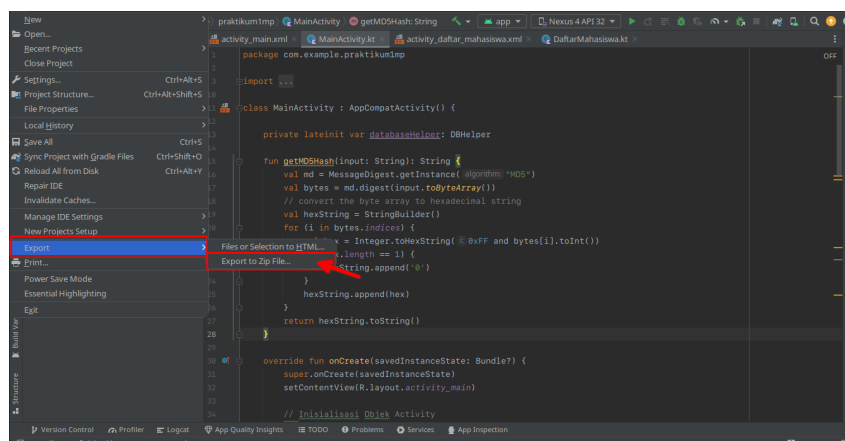
10. Lakukan import untuk **ContentValues** jika terjadi **error**
11. Jalankan **Emulator** terlebih dahulu, lalu buka jalankan aplikasi. Coba kirimkan data melalui form pendaftaran



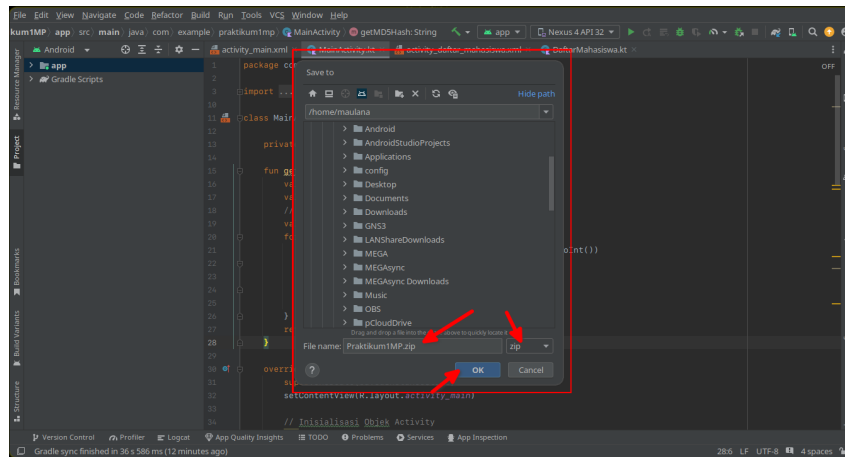
12. Jika kueri berhasil maka akan muncul **Toast** dan halaman akan kembali ke halaman **Login**



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 7

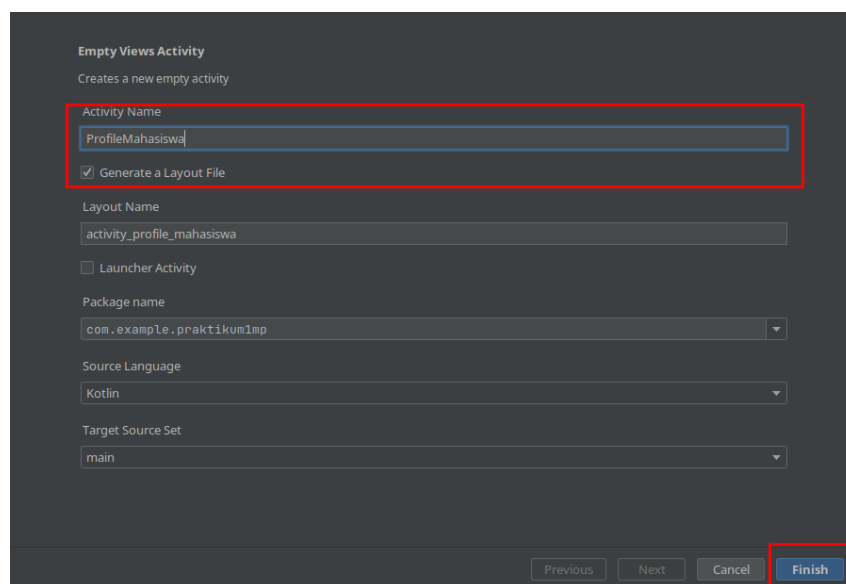
Praktikum 7

7.1 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

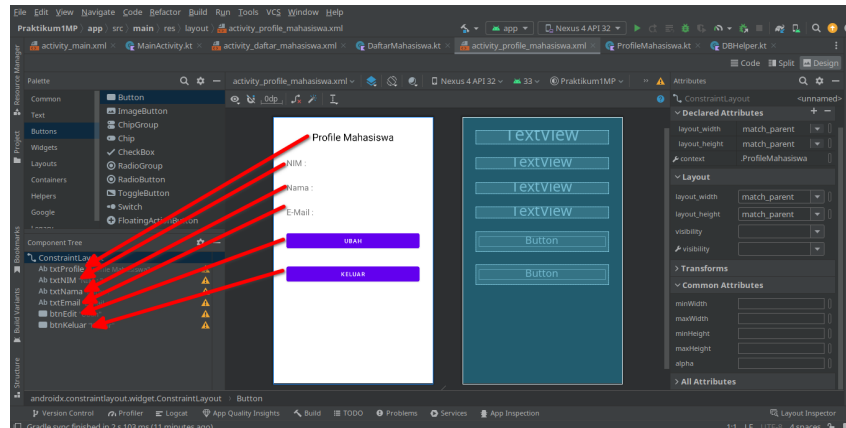
7.2 Tutorial

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

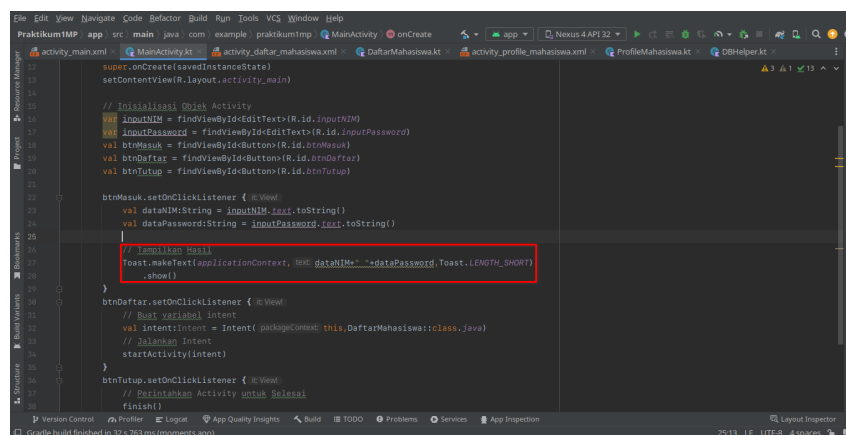


2. Buatlah tampilan halaman untuk **ProfileMahasiswa** di layout **activity_profile_mahasiswa.xml**. Perhatikan daftar objek berikut ini
 - TextView : **txtProfile** → Text : **Profile Mahasiswa**
 - TextView : **txtNIM** → Text : **NIM :**

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

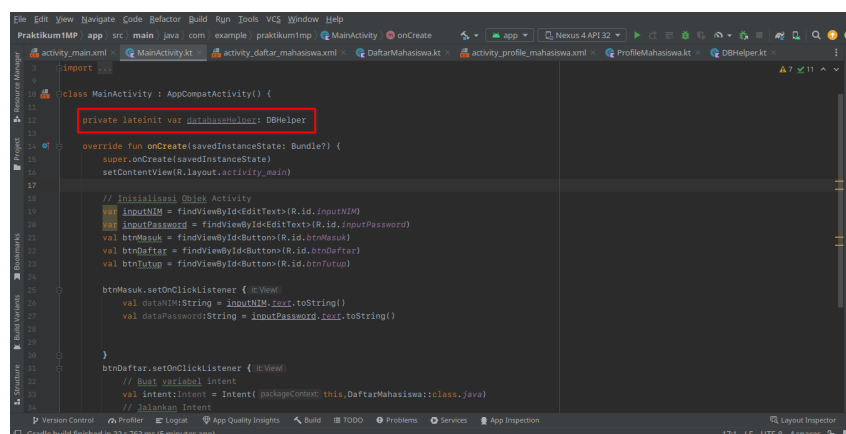


3. Sesudah layout untuk **ProfileMahasiswa** sudah jadi, berikutnya buka file **Main-Activity.kt** dan **hapus** bagian kode berikut



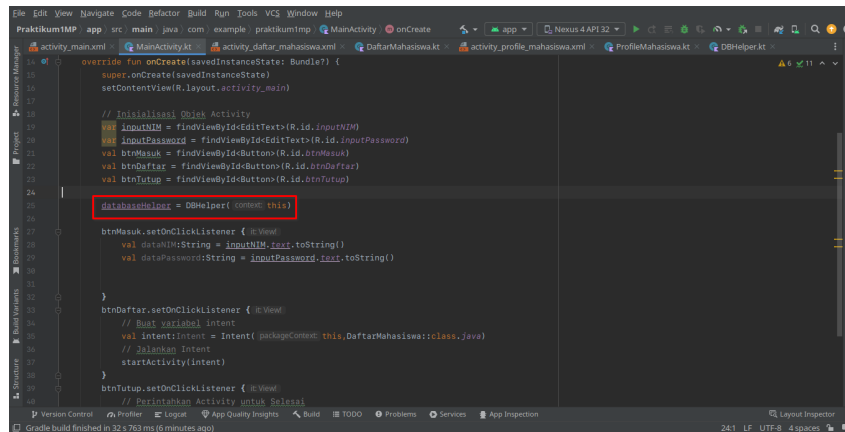
4. Sesudah dihapus, tambahkan **late init** Database di bawah baris **Class MainActivity**

Potongan Kode
private lateinit var databaseHelper: DBHelper



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

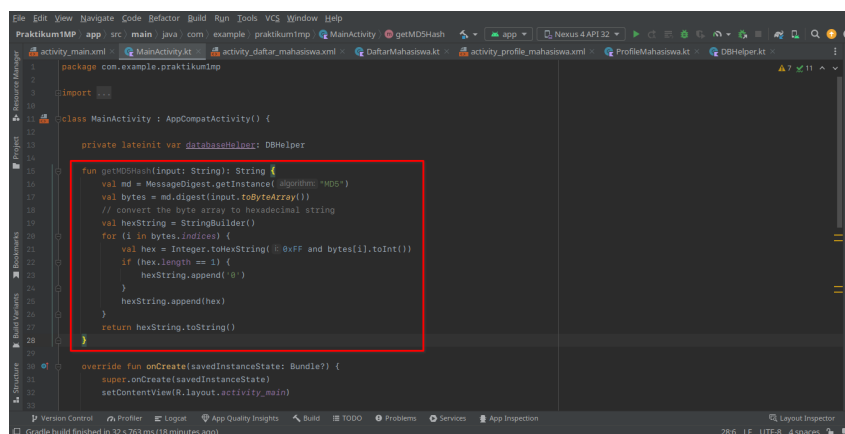
Potongan Kode
databaseHelper = DBHelper(this)



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

Potongan Kode

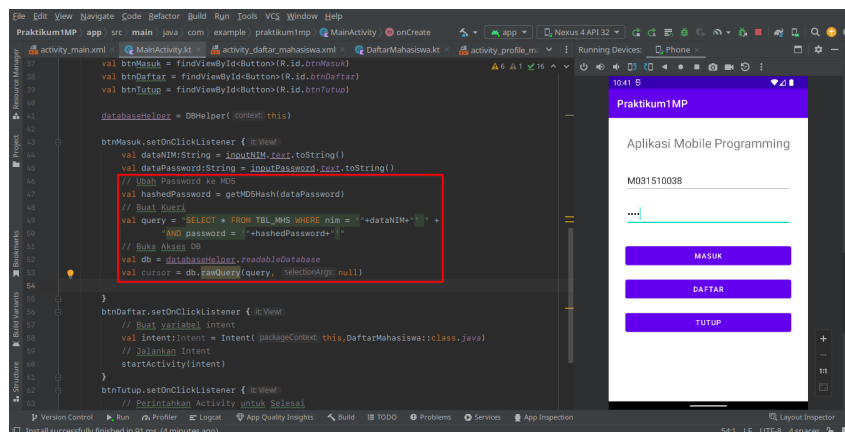
```
fun getMD5Hash(input: String): String {
    val md = MessageDigest.getInstance("MD5")
    val bytes = md.digest(input.toByteArray())
    // Konversi ke Hexa
    val hexString = StringBuilder()
    for (i in bytes.indices) {
        val hex = Integer.toHexString(0xFF and bytes[i].toInt())
       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



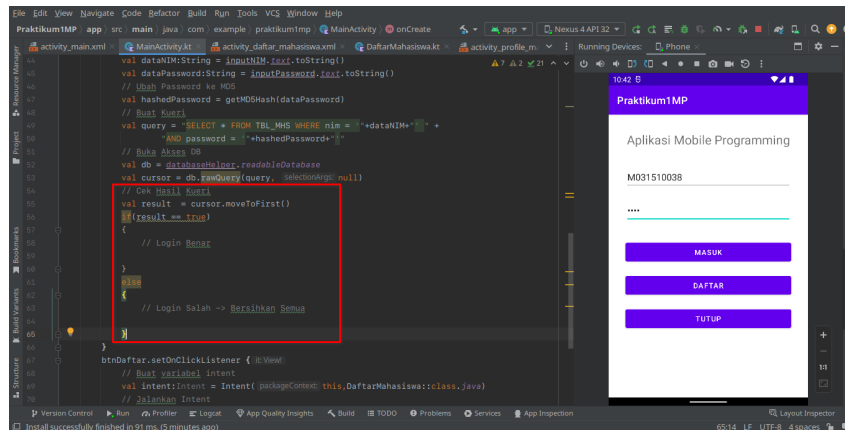
8. Kemudian lanjutkan kode untuk pengecekan hasil kueri, perhatikan kode dan gambar berikut:

Potongan Kode

```
// Cek Hasil Kueri
val result = cursor.moveToFirst()
if(result == true)
{
    // Login Benar

}
else
{
    // Login Salah -> Bersihkan Semua

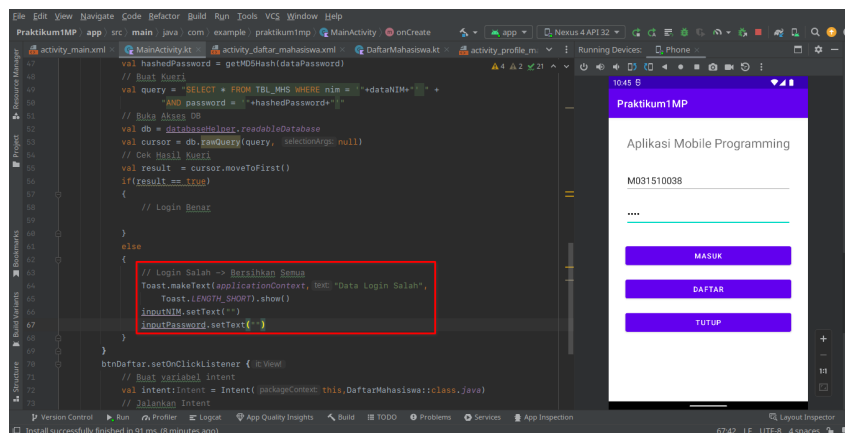
}
```



9. Ketika kode **IF** sudah dimasukkan sekarang, isi bagian **Login Salah** dengan kode berikut:

Potongan Kode

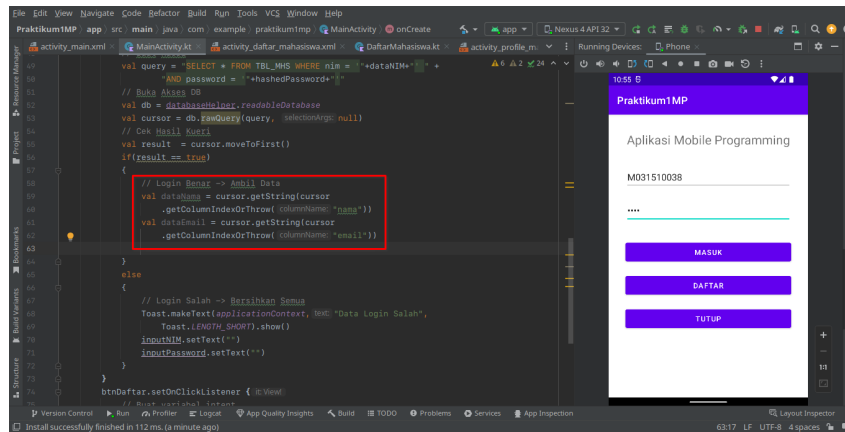
```
Toast.makeText(applicationContext,"Data Login Salah",
    Toast.LENGTH_SHORT).show()
inputNIM.setText("")
inputPassword.setText("")
```



10. Lalu masukkan kode untuk mengambil data untuk **Login Benar**

Potongan Kode

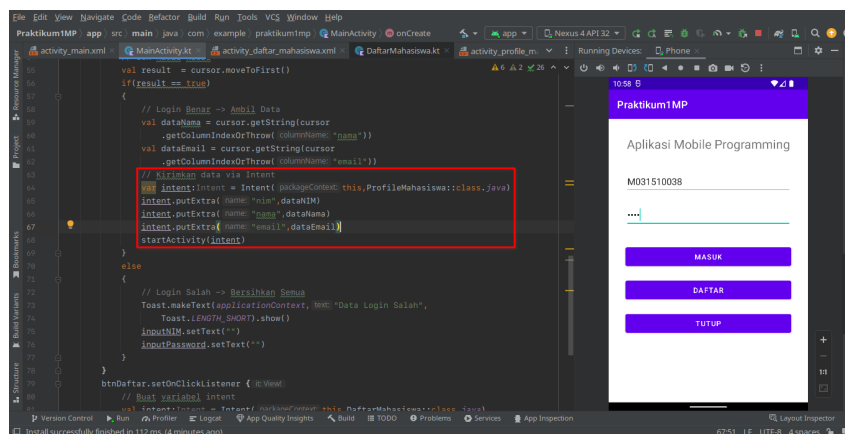
```
val dataNama = cursor.getString(cursor
    .getColumnIndexOrThrow("nama"))
val dataEmail = cursor.getString(cursor
    .getColumnIndexOrThrow("email"))
```



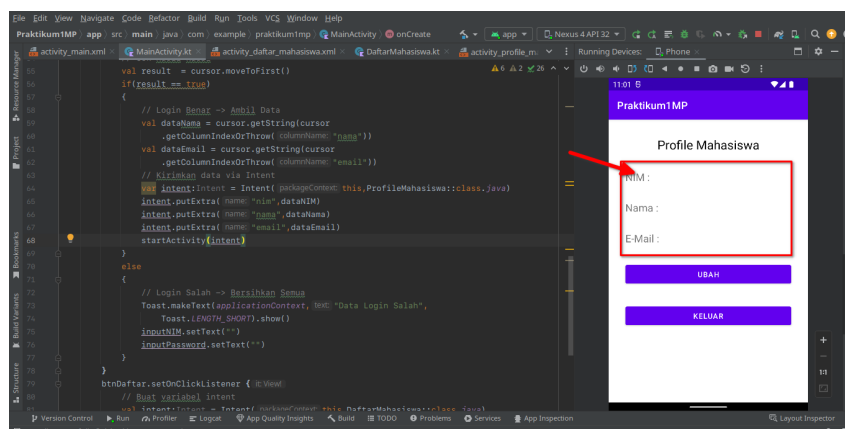
11. Setelah itu masukkan kode berikut untuk mengirimkan data-data tersebut ke halaman **ProfileMahasiswa**

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)
```



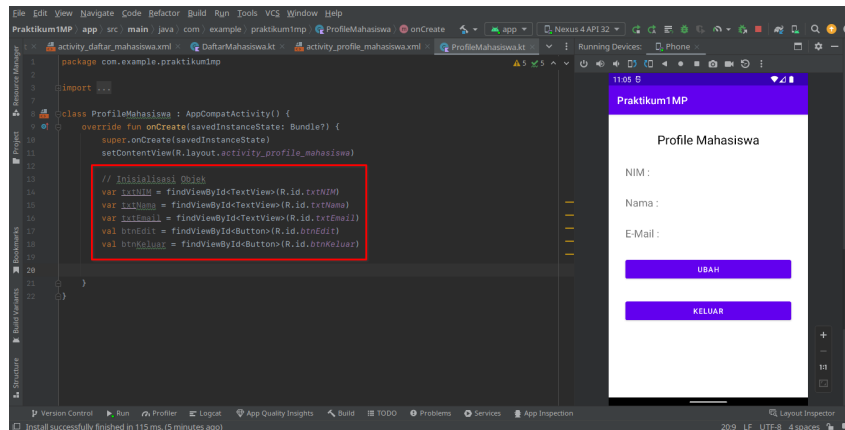
12. Jika dilakukan dengan benar, maka ketika Login aplikasi akan menampilkan Halaman Profile Mahasiswa. Tetapi masih kosong.



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

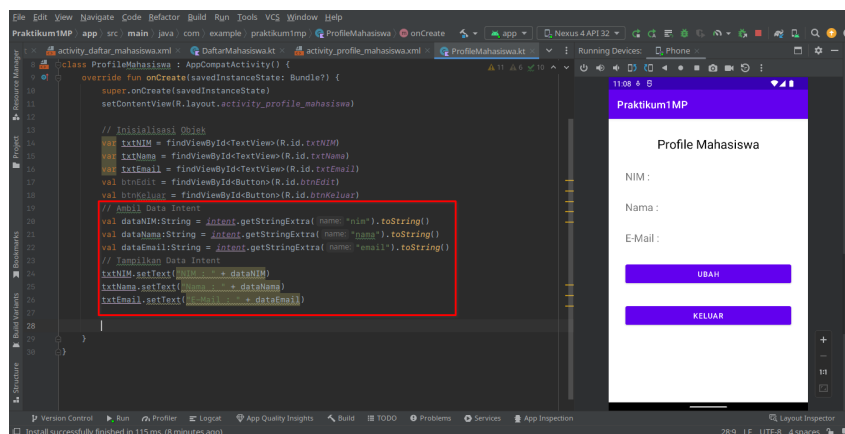
```
// Inisialisasi Objek
var txtNIM = findViewById<TextView>(R.id.txtNIM)
var txtNama = findViewById<TextView>(R.id.txtNama)
var txtEmail = findViewById<TextView>(R.id.txtEmail)
val btnEdit = findViewById<Button>(R.id.btnEdit)
val btnKeluar = findViewById<Button>(R.id.btnKeluar)
```



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)
```



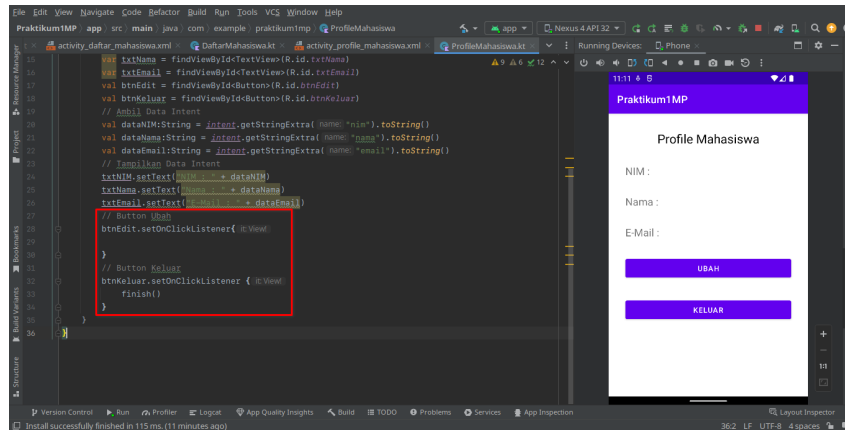
15. Terakhir tambahkan kode **Listener** untuk masing-masing **Button**

Potongan Kode

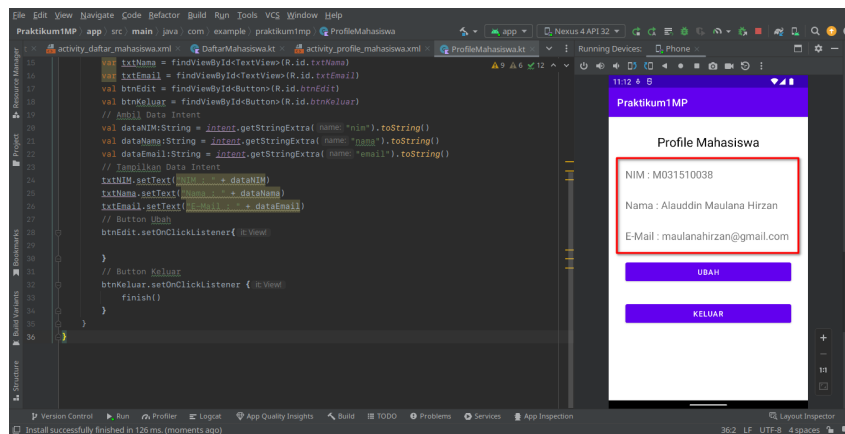
```
// Button Ubah
btnEdit.setOnClickListener{

}

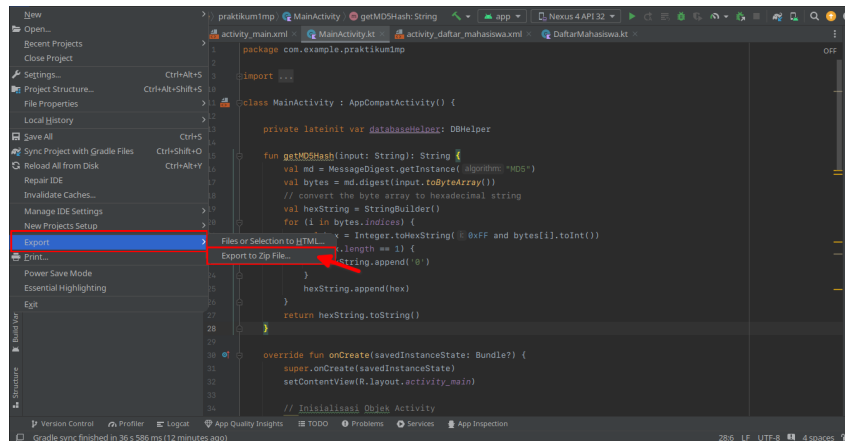
// Button Keluar
btnKeluar.setOnClickListener {
    finish()
}
```



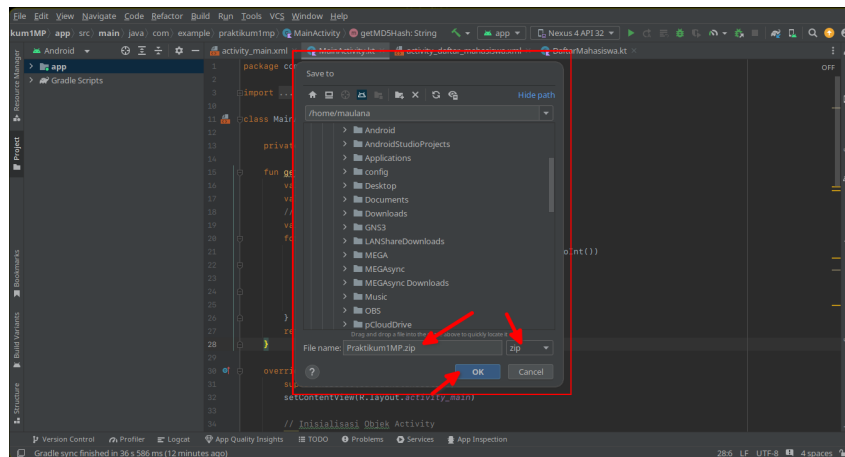
16. Nyalakan Emulator, Jalankan Aplikasi, dan Cobalah Login



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 8

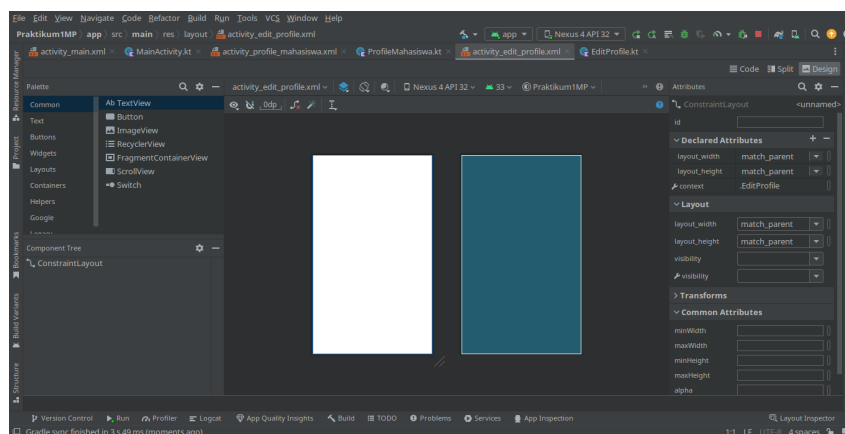
Praktikum 8

8.1 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

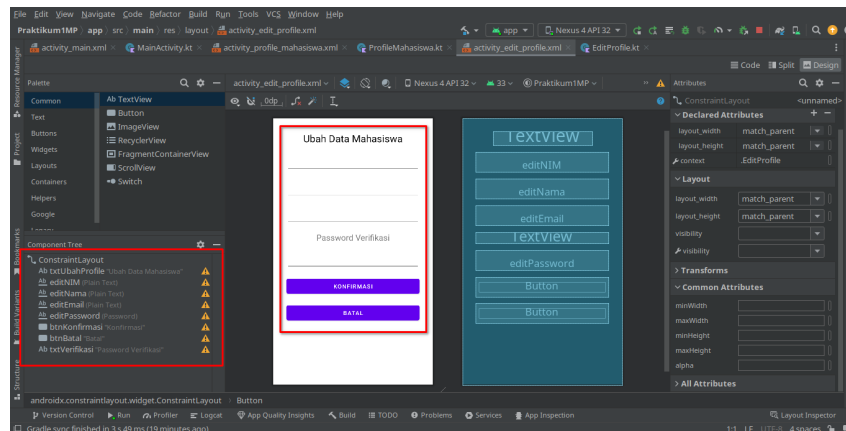
8.2 Tutorial

1. Buka Android Studio, dan buatlah satu **Activity Baru** dengan nama **EditProfile**.
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**

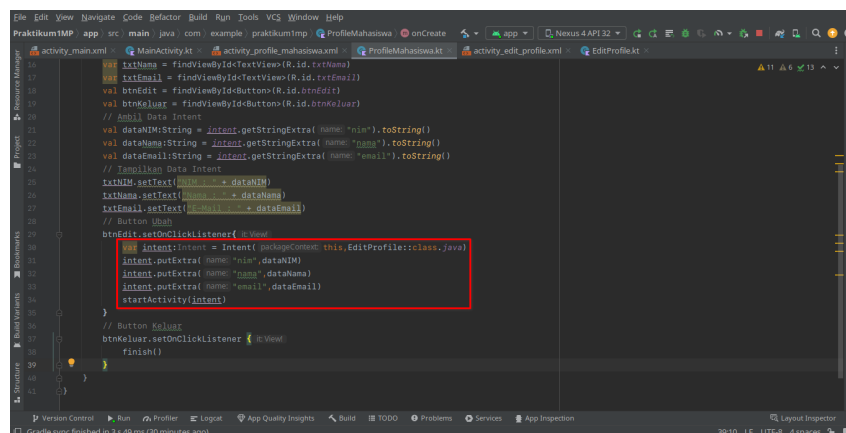
- EditText Password → **editPassword** : Text → (Kosong)
- Button → **btnKonfirmasi** : Text → **Konfirmasi**
- Button → **btnBatal** : Text → **Batal**



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

Potongan Kode

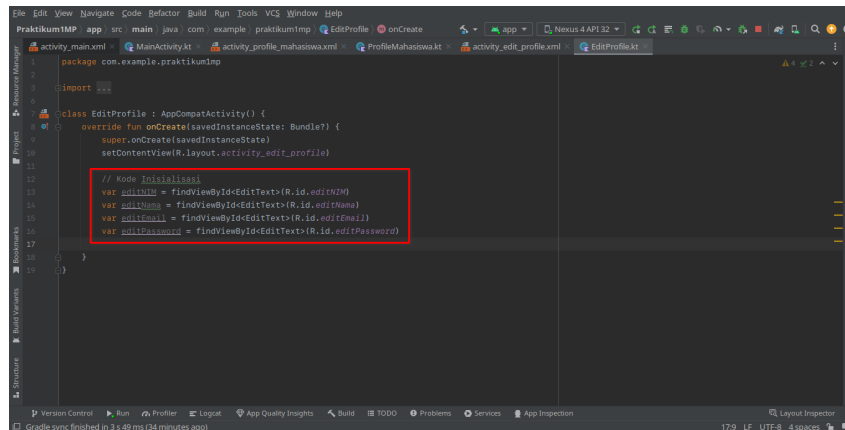
```
var intent:Intent = Intent(this,EditProfile::class.java)
intent.putExtra("nim",dataNIM)
intent.putExtra("nama",dataNama)
intent.putExtra("email",dataEmail)
startActivity(intent)
```



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

Potongan Kode

```
// Kode Inisialisasi
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)
```



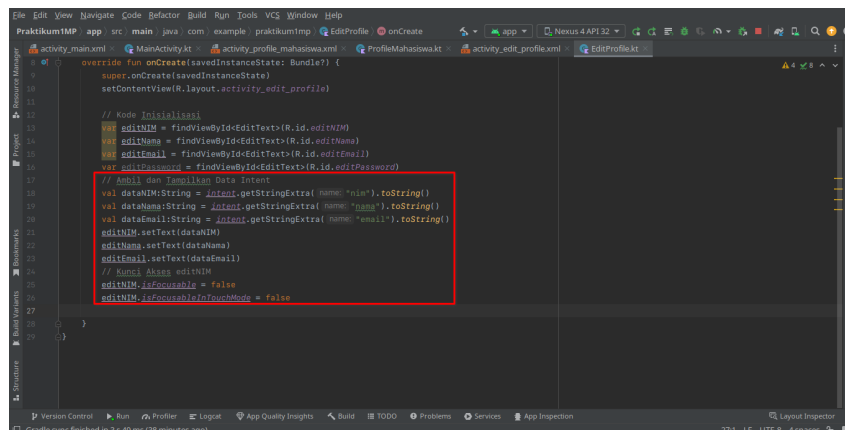
5. Tambahkan kode berikut untuk mengambil data dari Intent dan mengunci akses editNIM

Potongan Kode

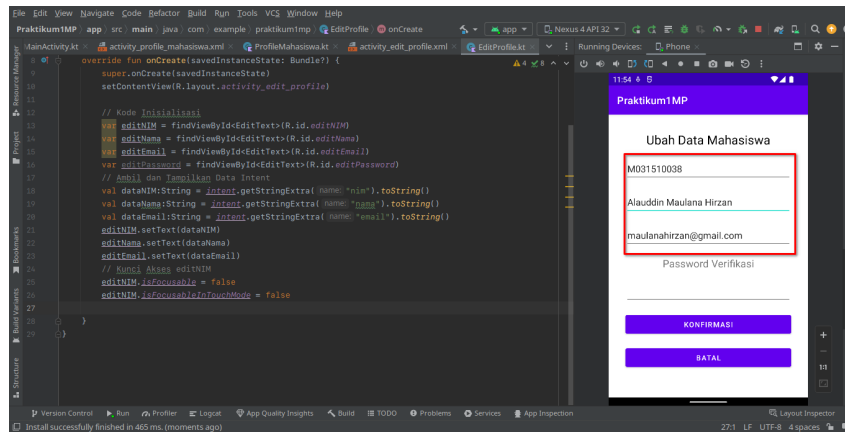
```

// 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

```



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



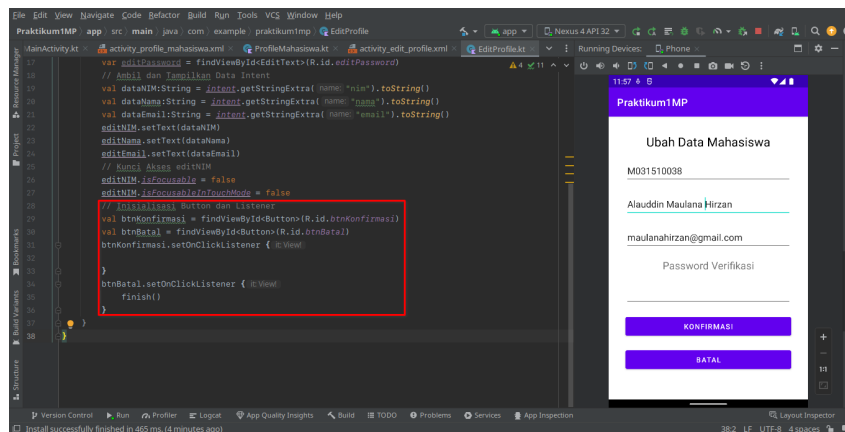
7. Berikutnya tambahkan kode inialisasi Button dan fungsi Listener di masing-masing tombol. Perhatikan kode berikut

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()
}
```

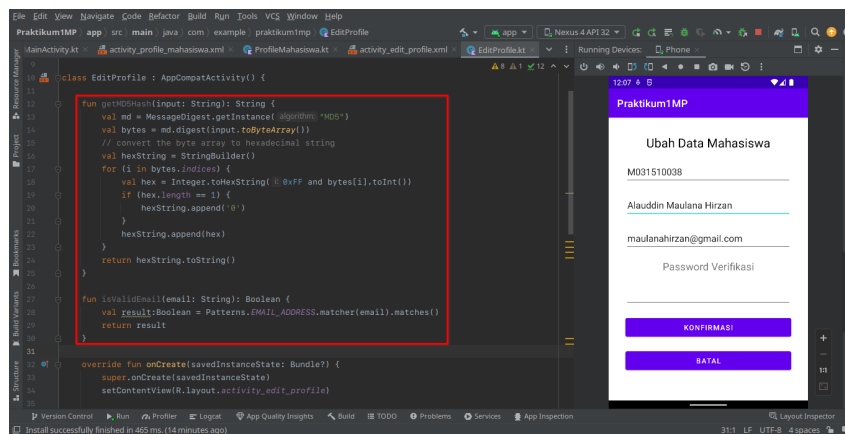


8. Berikutnya masukkan kode untuk mendukung perubahan data seperti **Hashing MD5** dan **E-Mail Checker**. Letakkan dibawah baris kode **Class**. Perhatikan contoh berikut:

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()
}

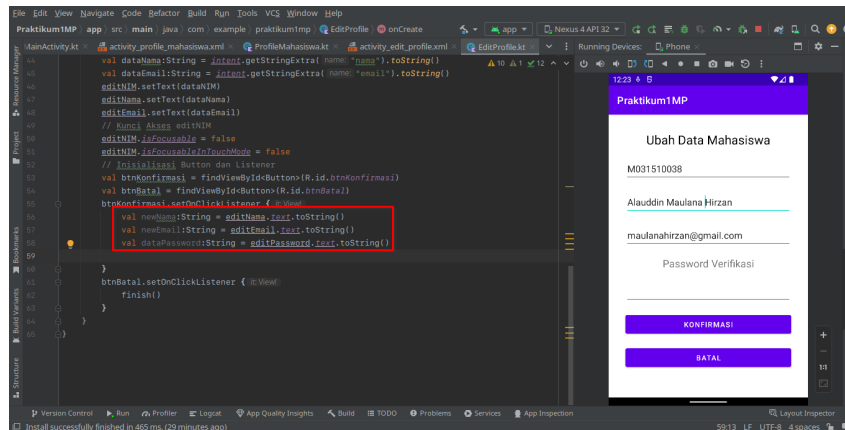
fun isValidEmail(email: String): Boolean {
    val result: Boolean = Patterns.EMAIL_ADDRESS.matcher(email).matches()
    return result
}
```



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()
```

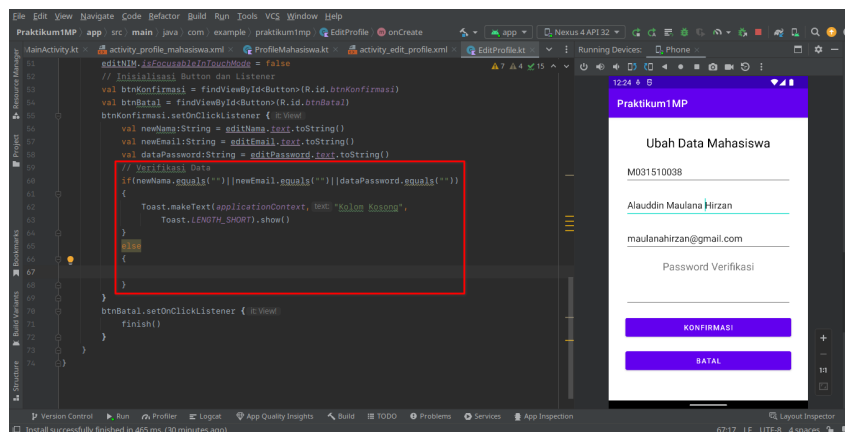
10. Lalu tambahkan kode untuk verifikasi data baru. Perhatikan contoh:

Potongan Kode

```

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

```



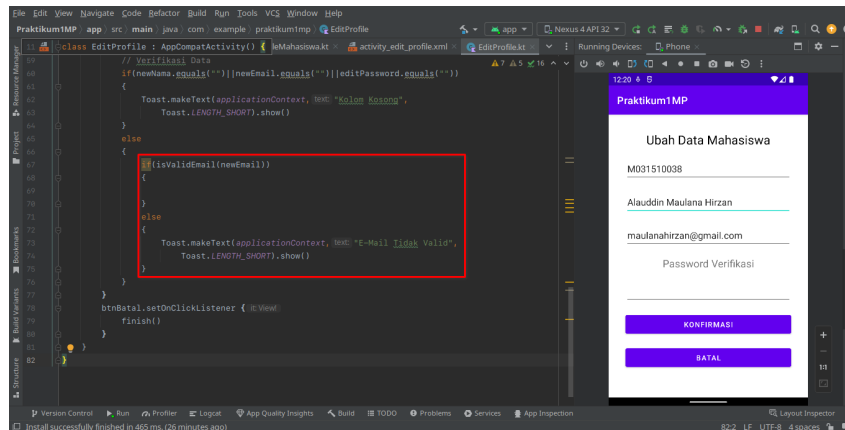
11. Kemudian tambahkan kode verifikasi E-Mail tepat di dalam **ELSE**

Potongan Kode

```

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

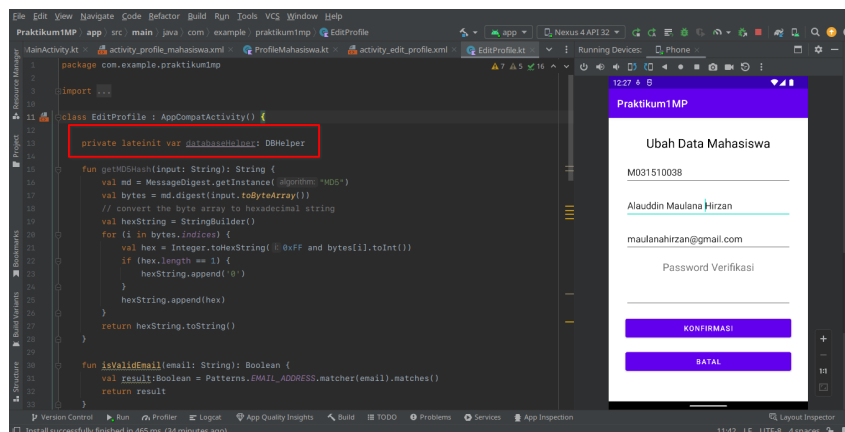
```



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

Potongan Kode

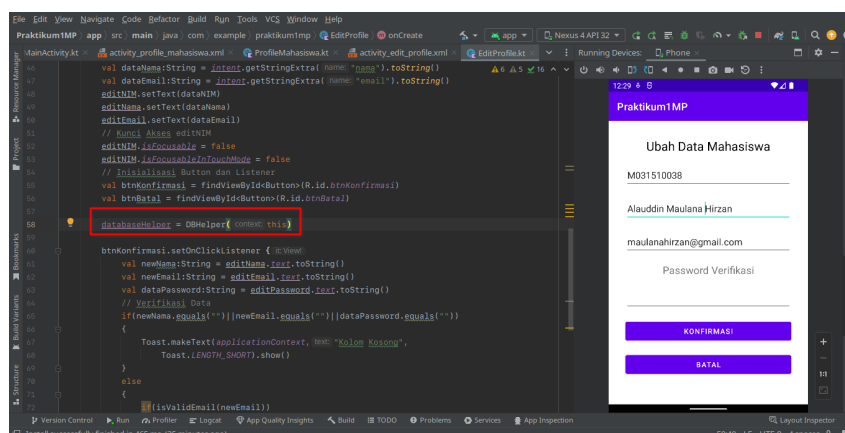
private lateinit var databaseHelper: DBHelper



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

Potongan Kode

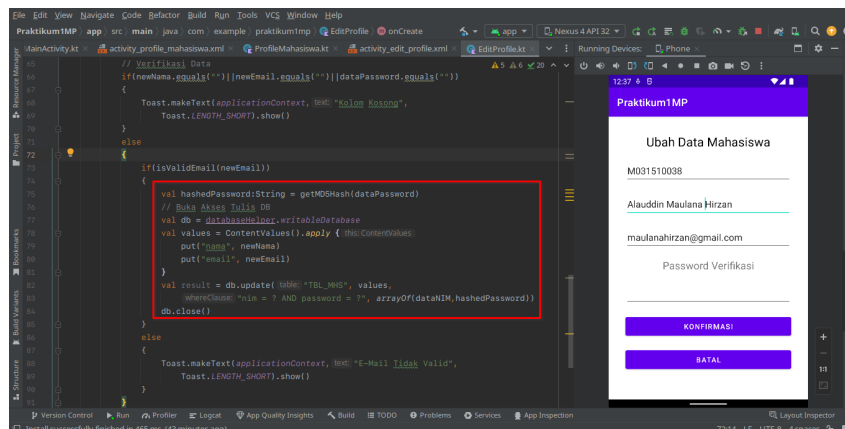
databaseHelper = DBHelper(this)



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

Potongan Kode

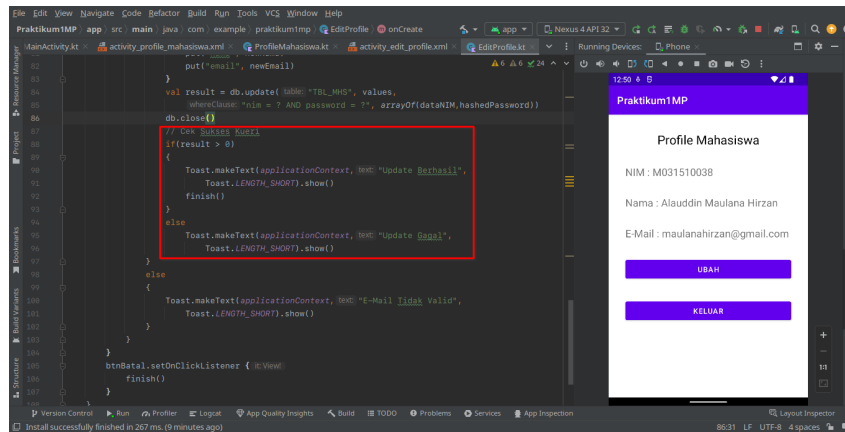
```
val hashedPassword:String = getMD5Hash(dataPassword)
// Buka Akses Tulis DB
val db = databaseHelper.writableDatabase
val values = ContentValues().apply {
    put("nama", newNama)
    put("email", newEmail)
}
val result = db.update("TBL_MHS", values,
    "nim = ? AND password = ?", arrayOf(dataNIM,hashedPassword))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



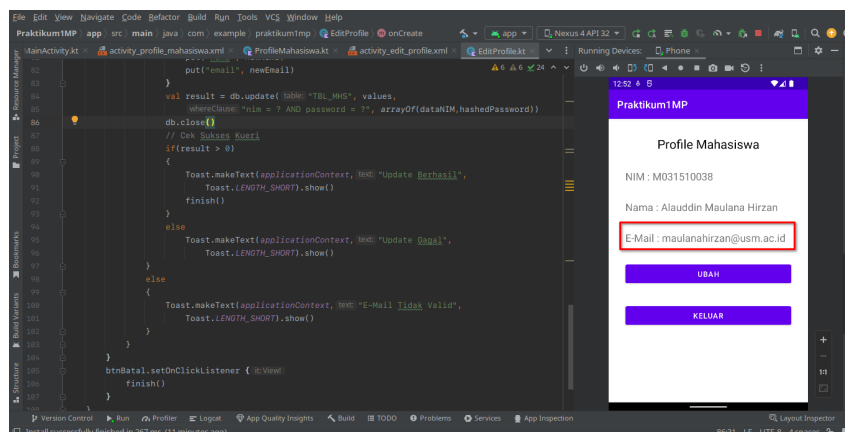
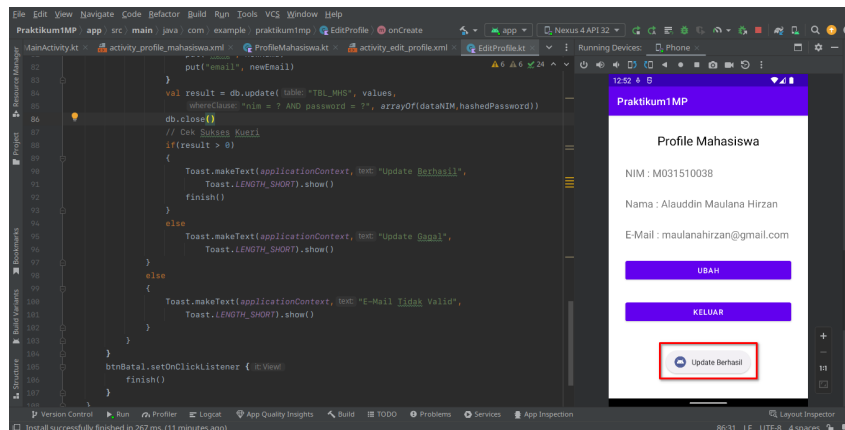
15. Terakhir tambahkan kode pengecekan update tepat di bawah kode sebelumnya. Perhatikan contoh:

Potongan Kode

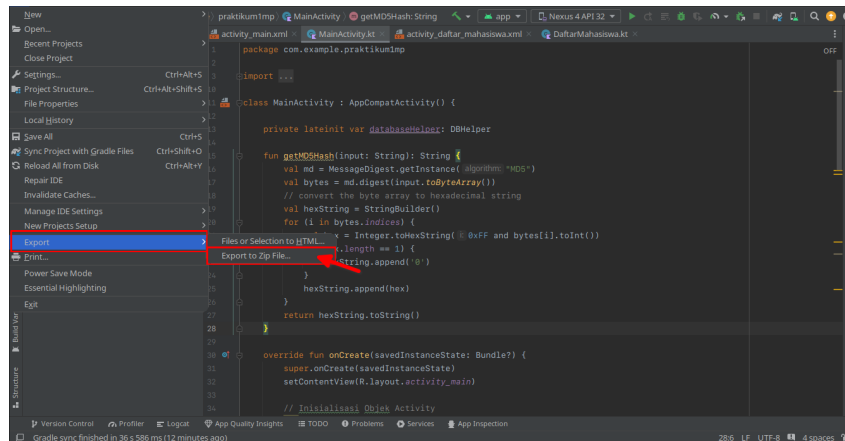
```
// Cek Sukses Kueri
if(result > 0)
{
    Toast.makeText(applicationContext,"Update Berhasil",
        Toast.LENGTH_SHORT).show()
    finish()
}
else
    Toast.makeText(applicationContext,"Update Gagal",
        Toast.LENGTH_SHORT).show()
```



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

