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

Daftar Gambar

Pendahuluan

0.1 Mengenal 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 Mengenal 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

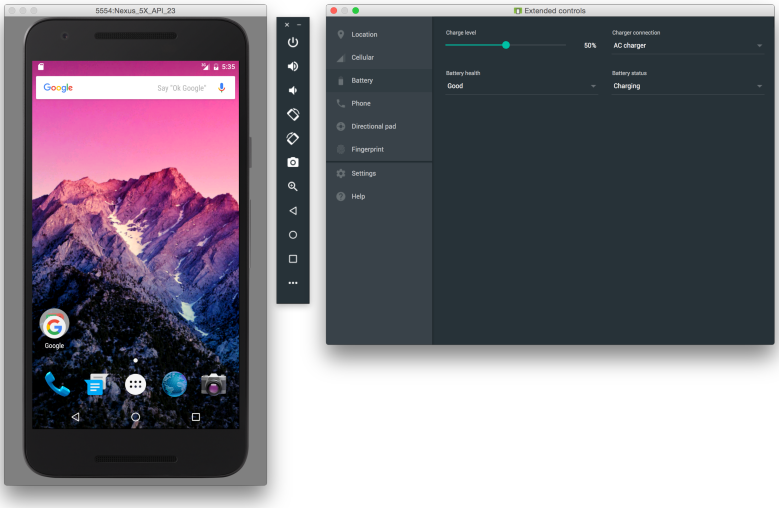
0.3 Mengenal Android SDK

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 Mengenal 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 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://developer.android.com/studio/run/win-usb

Bab 1

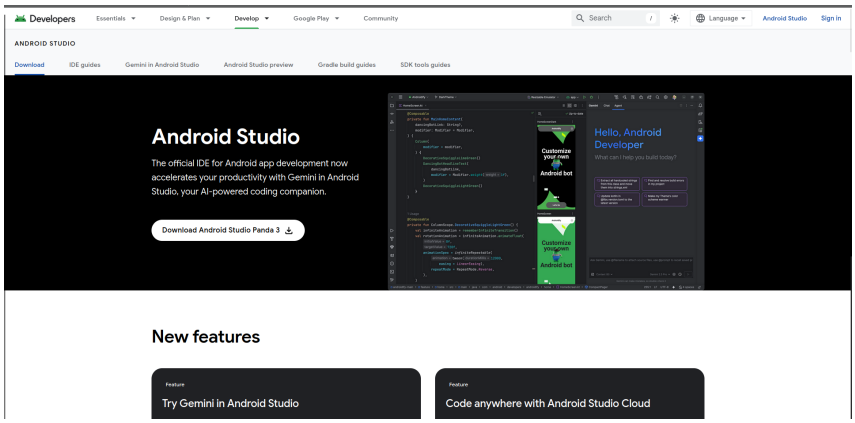
Instalasi Android Studio, SDK, dan Emulator

1.1 Pendahuluan

Pada pertemuan ini, praktikan akan mempelajari dan melakukan proses instalasi serta konfigurasi lingkungan pengembangan aplikasi Android menggunakan Android Studio. Kegiatan dimulai dengan instalasi Android Studio, dilanjutkan dengan pengunduhan dan pengaturan Android SDK sebagai komponen utama dalam pengembangan aplikasi. Selanjutnya, praktikan akan mengonfigurasi emulator menggunakan Android Virtual Device (AVD) untuk mensimulasikan perangkat Android secara virtual. Tahap ini mencakup pembuatan device profile, pemilihan versi sistem operasi, serta pengaturan performa emulator. Setelah lingkungan siap, mahasiswa akan membuat project Android pertama menggunakan template dasar yang disediakan Android Studio. Proses ini meliputi pemilihan jenis activity, pengaturan nama project, serta struktur awal aplikasi. Sebagai hasil akhir pertemuan, mahasiswa diharapkan mampu menjalankan project Android pertama pada emulator tanpa error, sehingga memastikan bahwa seluruh lingkungan pengembangan telah terkonfigurasi dengan benar.

1.2 Tutorial

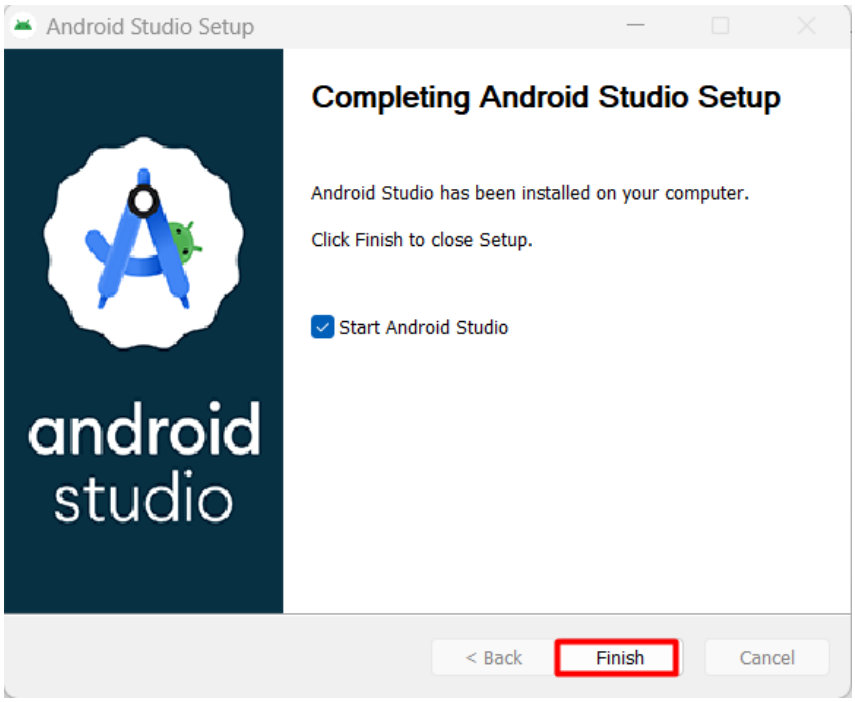
1. Mahasiswa mengunduh Android Studio terbaru menggunakan link https://developer.android.com/studio. Pastikan untuh mengunduh versi terbaru.



Gambar 1.1: Installer Android Studio

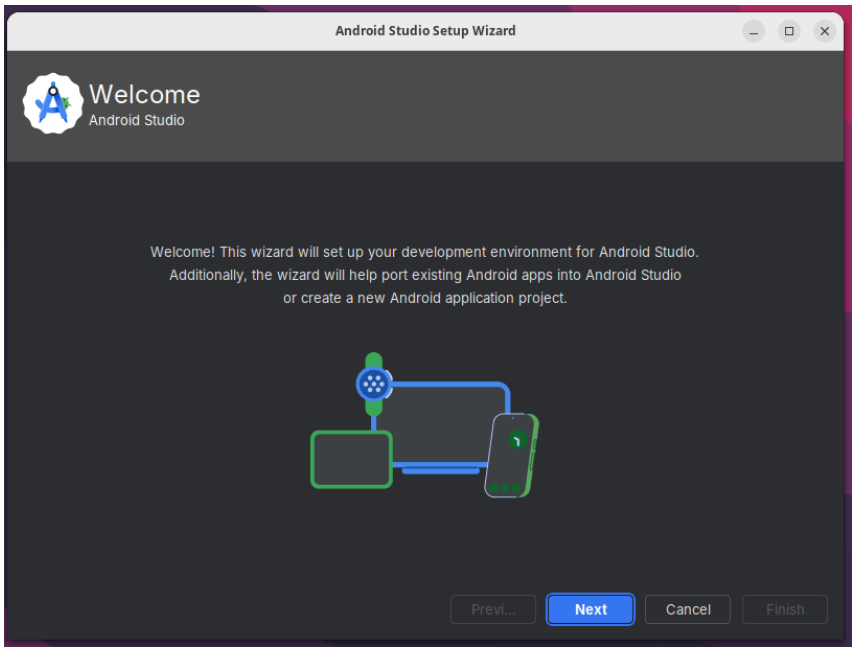
2. Jalankan *Installer* sesuai Sistem Operasi masing-masing hingga finish. Cek video panduan berikut: Klik Link Ini

3. Setelah instalasi selesai seperti gambar berikut. Klik Finish

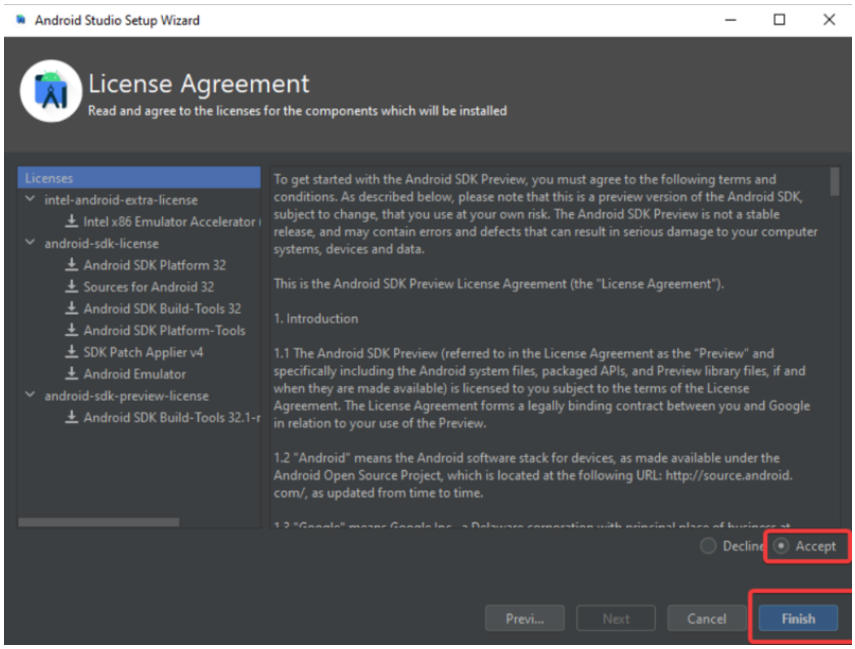


Gambar 1.2: Instalasi Selesai

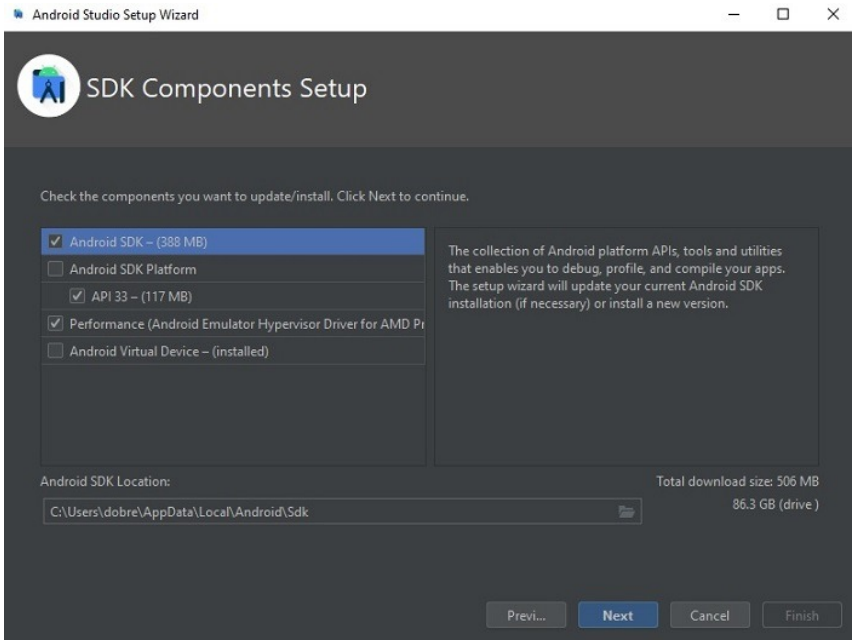
4. Android Studio telah dipasang dengan baik. Berikutnya adalah konfigurasi SDK dan Emulator untuk Android. Proses berjalan secara otomatis, dan mahasiswa cukup menyetujui perjanjian yang ada.



Gambar 1.3: Android Studio Wizard

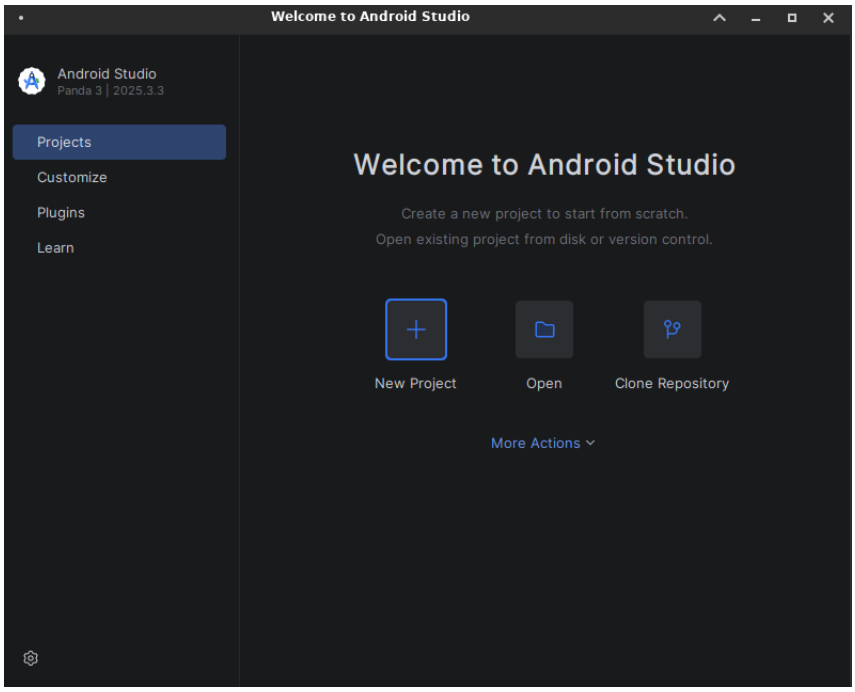


Gambar 1.4: Perjanjian Android Studio



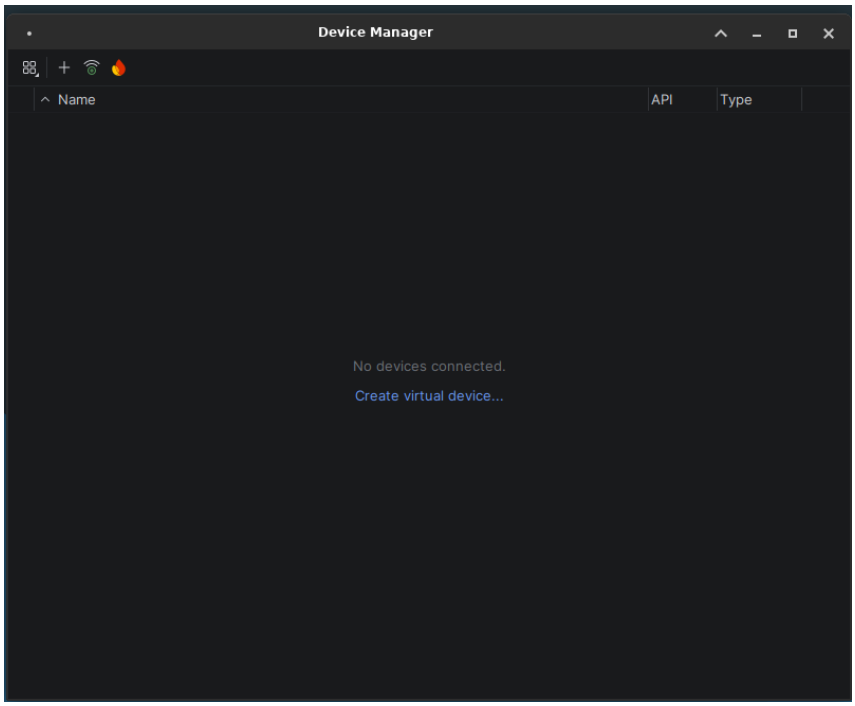
Gambar 1.5: Instalasi SDK

5. JIka semua instalasi dasar sudah terpenuh semua, dan Android Studio menampilkan *Welcome Page*, maka bisa melanjutkan ke langkah berikutnya untuk menambahkan emulator



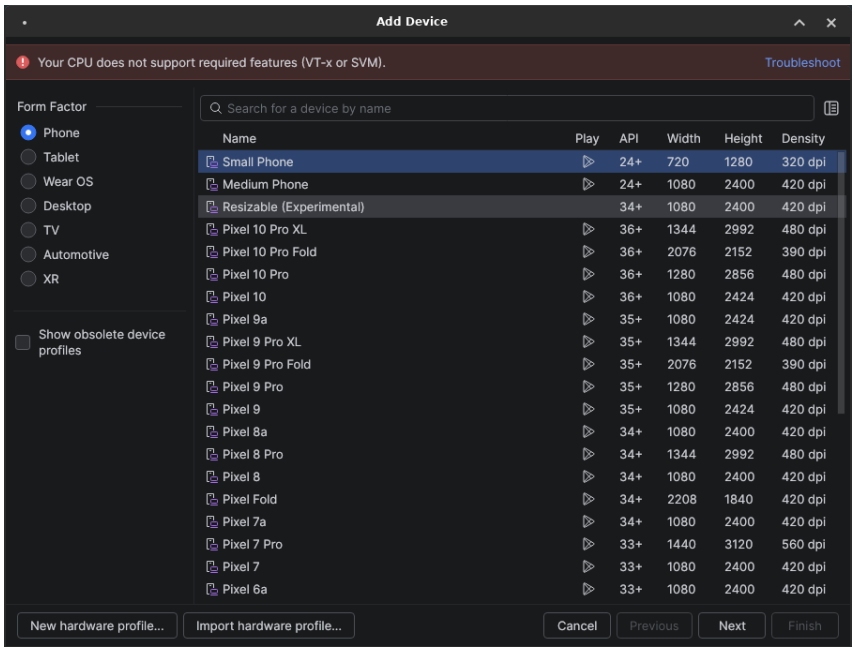
Gambar 1.6: Welcome Page

6. Untuk menambahkan Emulator dari halaman tersebut. Klik *More Actions* atau *Titik Tiga*, pilih *Virtual Device Manager*. Maka akan muncul halaman berikut



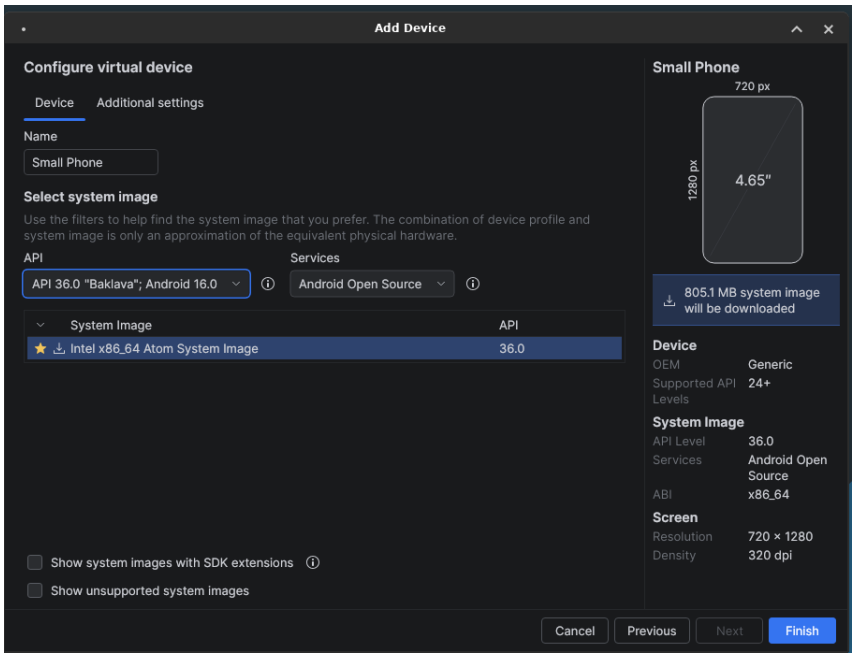
Gambar 1.7: Menambah Emulator

7. Klik *Create virtual device*, dan pilih template perangkat keras yang akan digunakan. Untuk menghemat RAM, pilih template *Small Phone*. Lalu klik *Next*



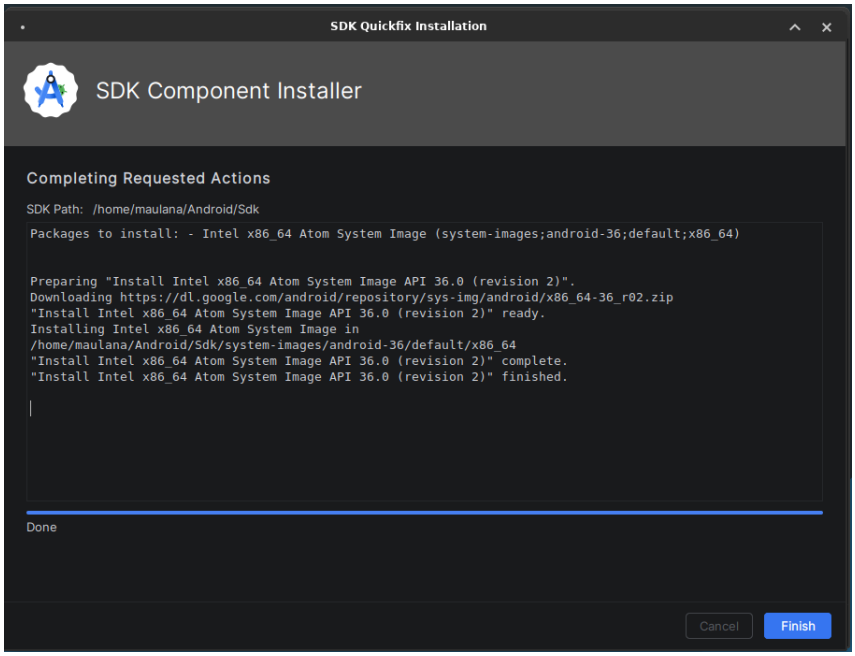
Gambar 1.8: Pemilihan Template Emulator

8. Untuk mempercepat proses *Download*, pilih *API 36.0 Baklava* dengan Service *Android Open Source*. Pilih *System Image* : `Intel X86_64 Atom System Image`. Lalu klik *Finish*



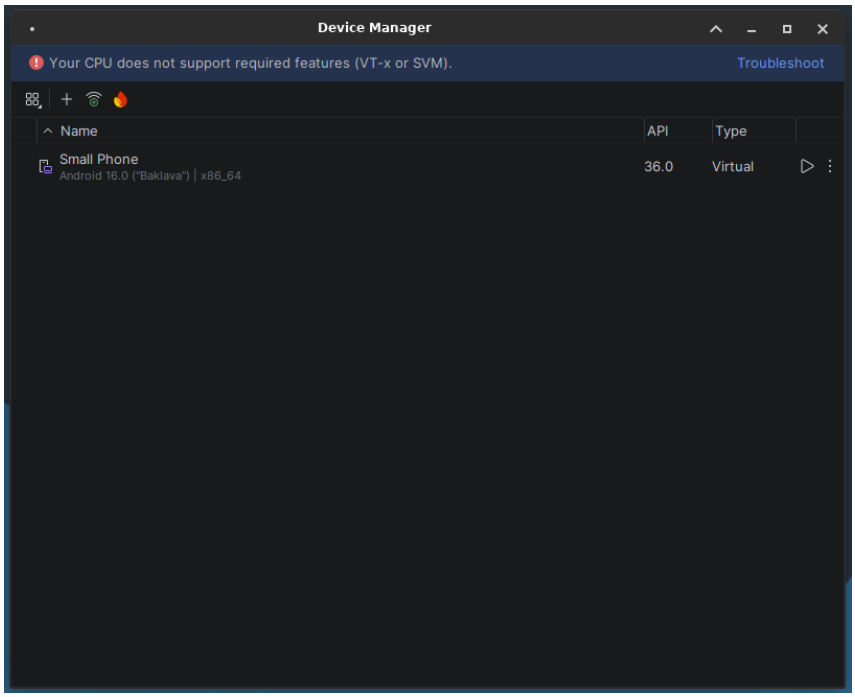
Gambar 1.9: Pemilihan API dan Service Android

9. Konfirmasi Download, dan tunggu hingga proses selesai.



Gambar 1.10: Proses Download System Image

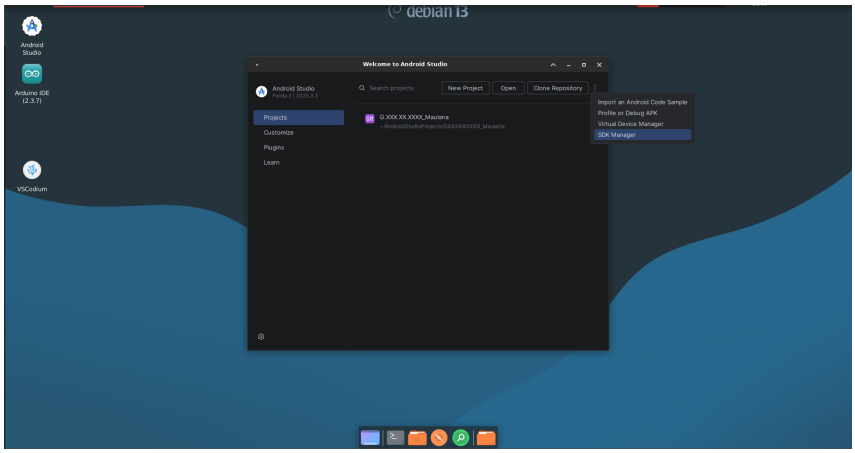
10. Jika berhasil, maka emulator baru akan ditampilkan



Gambar 1.11: Emulator Sukses Ditambahkan

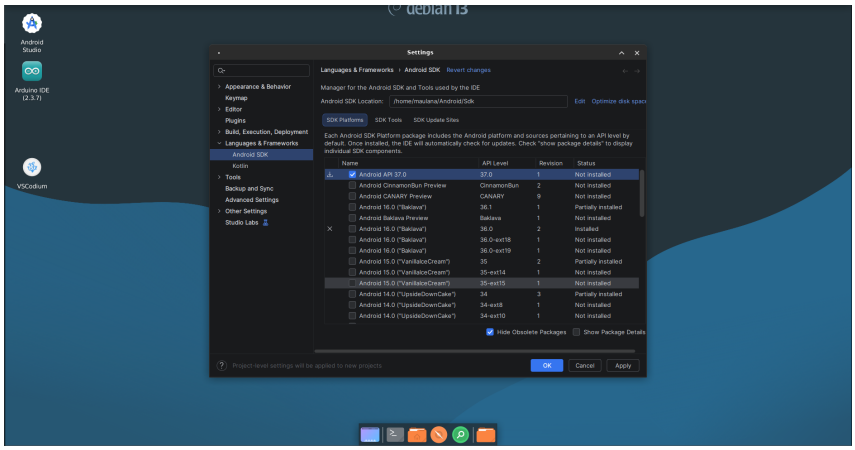
11. Alternatif menggunakan perangkat keras sendiri bisa melalui tutorial link berikut: Tutorial ADB

12. Setelah selesai membuat emulator, berikutnya adalah menginstal SDK. Tekan *More Actions* atau *Tiga Titik*, pilih *SDK Manager*.

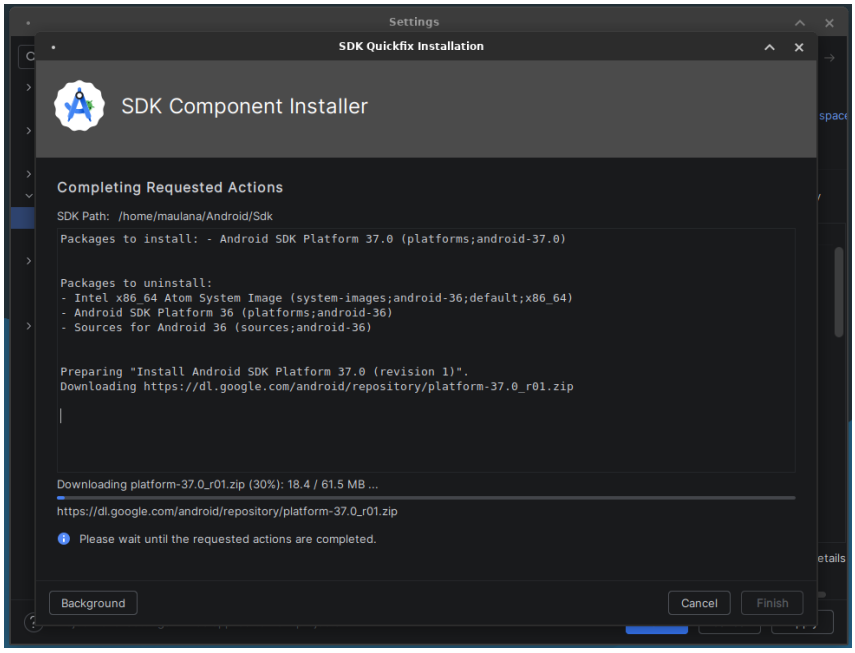


Gambar 1.12: Mengakses SDK Manager

13. Android Studio akan membuka halaman baru, pastikan pilih SDK terbaru tapi tidak memiliki kata *Preview*. Contoh *Android API 37*. Hilangkan semua centang, kecuali SDK terbaru. Klik OK. Android Studio akan menginstall SDK baru tersebut. Konfirmasi perubahan, dan Android Studio akan mengunduh SDK terbaru.

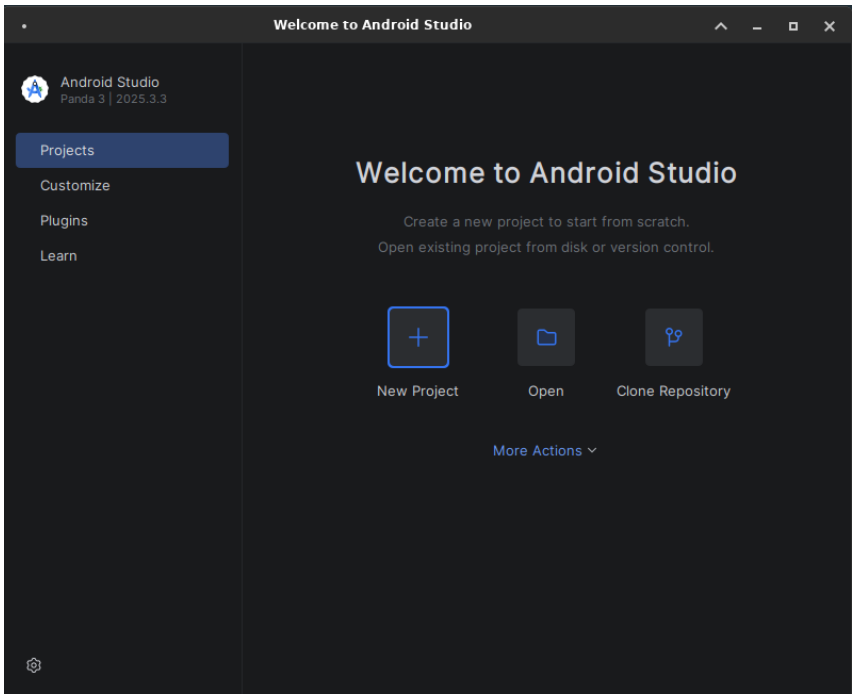


Gambar 1.13: Memilih SDK



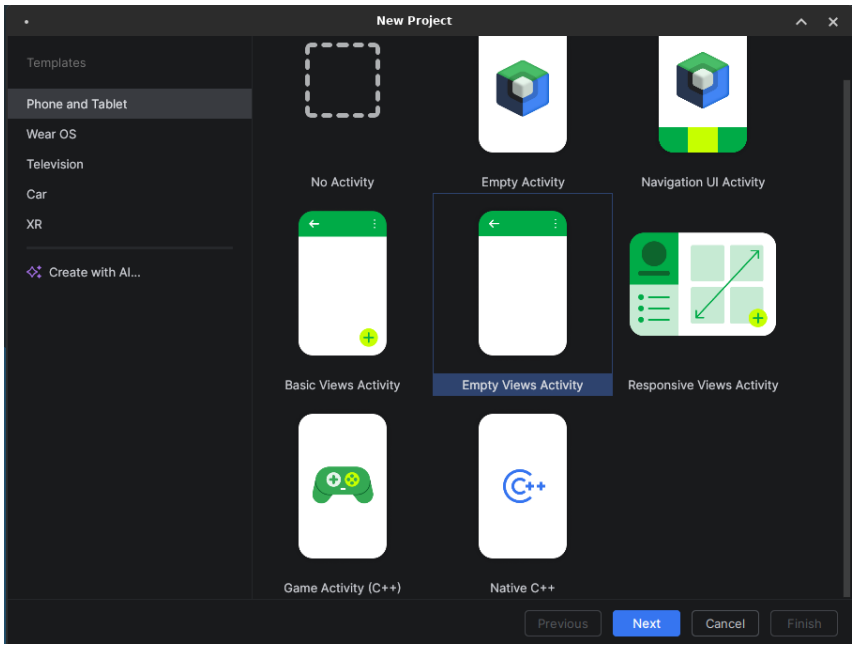
Gambar 1.14: Mengunduh SDK

14. Kembali ke halaman *Welcome to Android Studio*. Buat projek baru dengan klik *New Project*



Gambar 1.15: Membuat Projek Baru

15. Berikutnya Android Studio akan menampilkan pemilihan layout. Pilih *Empty View Activity* dengan ciri khas header hijau dan body putih tanpa logo. Klik *Next* untuk melanjutkan proses berikut nya

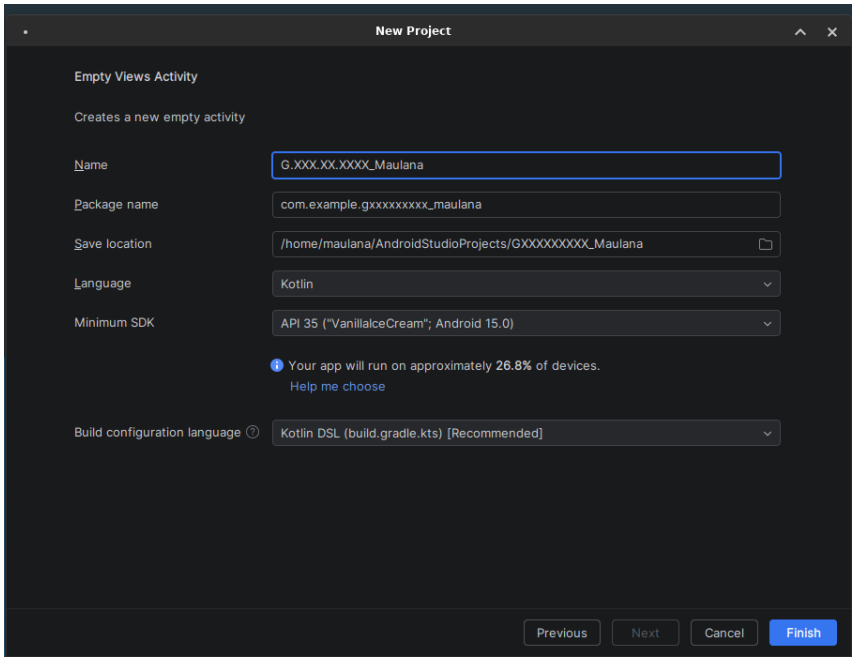


Gambar 1.16: Pemilihan Layout Empty View Activity

16. Setelah memilih layout dasar aplikasi, berikutnya adalah mengatur nama aplikasi, lokasi aplikasi, dan minimum SDK yang digunakan. Klik /Finish/ dan tunggu hingga Android Studio benar-benar selesai.

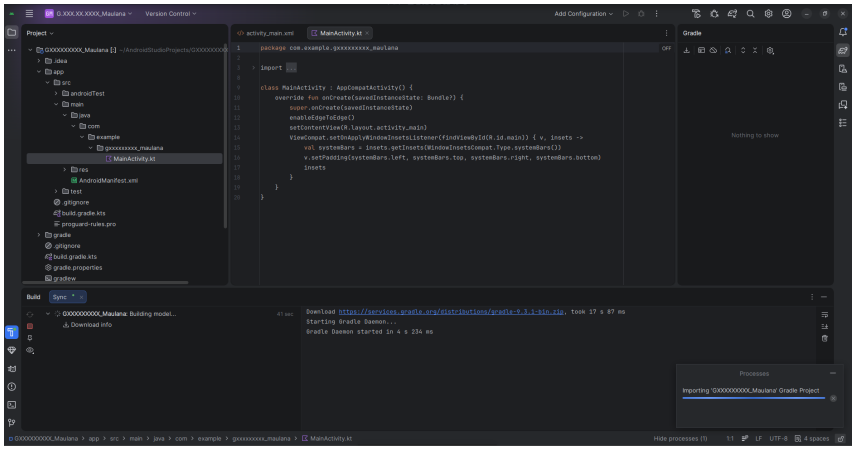
    - Untuk nama aplikasi, gunakan nim dan nama agar membedakan dengan mahasiswa lain.

- Untuk minimum SDK, gunakan Versi Android Perangkat/Emulator (Contoh 16) dikurangi 1. Maka Minimum SDK adalah 15



Gambar 1.17: Konfigurasi Dasar Aplikasi

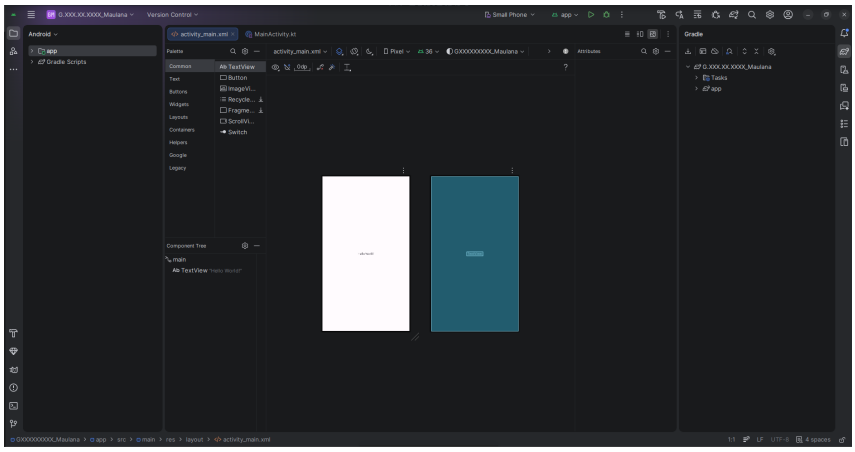
17. Android Studio akan menyiapkan proyek yang cukup membutuhkan waktu. Tergantung kecepatan internet, akan mempengaruhi lama konfigurasi awal proyek.



Gambar 1.18: Android Studio Menyiapkan Proyek

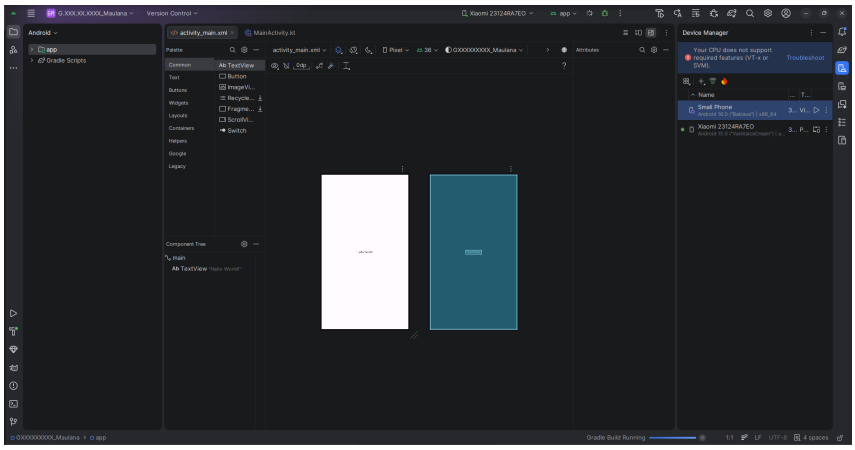
18. Jika Android meminta *Exclude Folder*, klik *Exclude* untuk mengamankan proyek dari Windows Defender. **Hanya Berlaku Windows**

19. Android Studio dikatakan selesai dan siap digunakan ketika *progress bar* di bawah selesai dan Layout Pertama di file `activity_main.xml` muncul dan bukan berupa kode xml



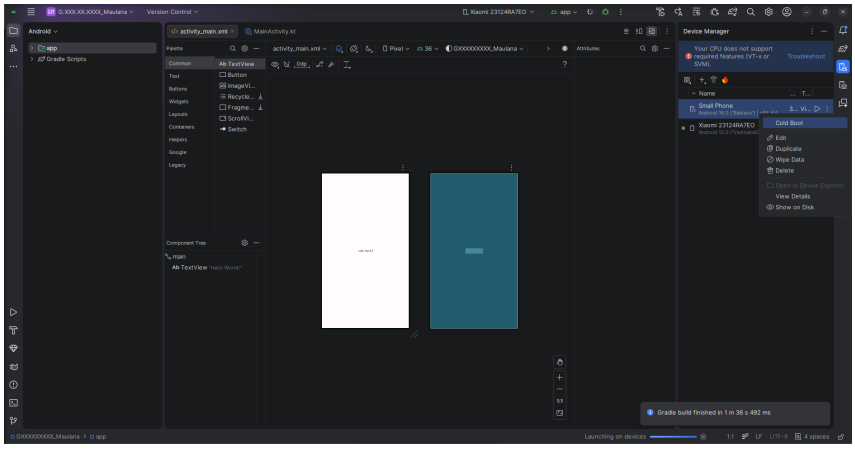
Gambar 1.19: Android Studio Siap Digunakan

20. Untuk menjalankan pertama kali aplikasi, di bagian sisi paling kanan pilih *Device Manager*.



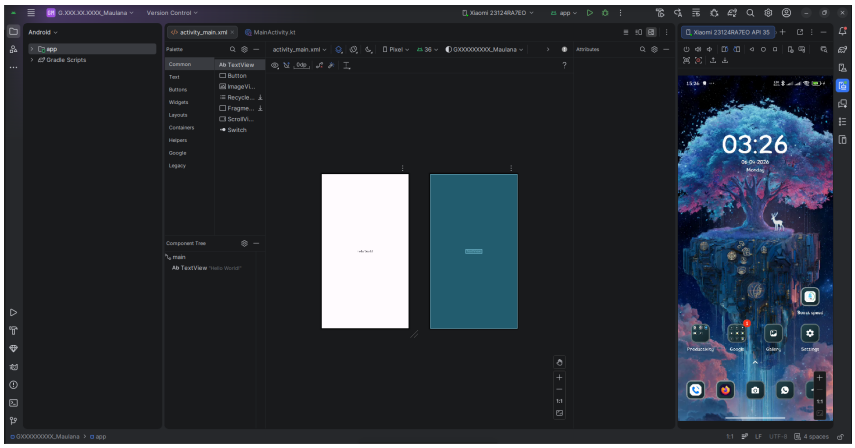
Gambar 1.20: Panel Device Manager

21. Pada emulator *Small Phone*, pilih *Cold Boot*. Lakukan cara ini untuk tidak memberatkan RAM.



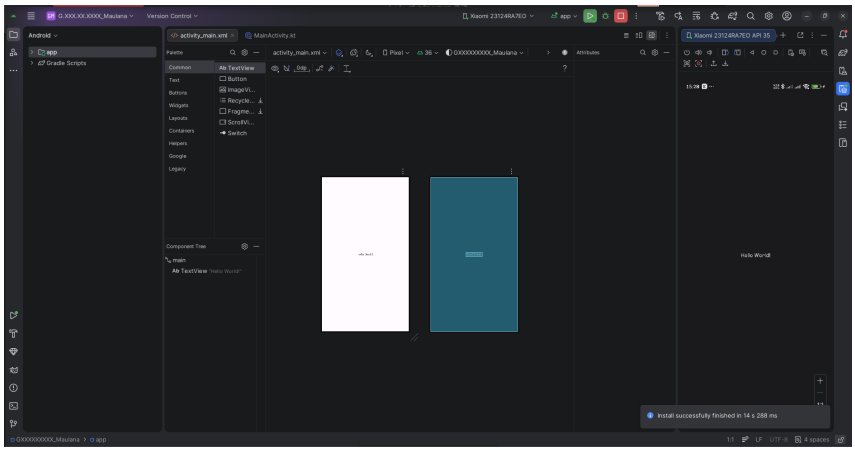
Gambar 1.21: Panel Device Manager

22. Tunggu hingga muncul tampilan Android



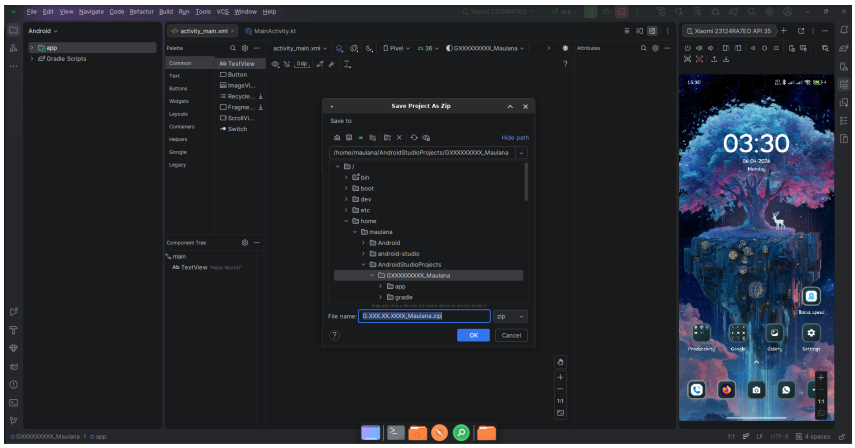
Gambar 1.22: Tampilan Android Emulator

23. Pastikan di bagian atas sudah merujuk ke emulator / smartphone, sebelum menekan tombol *Play (run app)*. Jika sudah klik *Play*, tunggu beberapa saat hingga muncul di emulator.



Gambar 1.23: Aplikasi Sukses Terpasang

24. Untuk mengirimkan proyek ini ke *e-learning*, klik *Empat Strip* sebelah kiri atas. Pilih *Export* dan pilih *Export to ZIP file*. Pilih lokasi yang aman, masukkan `NIM_Nama.zip` lalu tekan *OK*



Gambar 1.24: Ekspor Proyek ke ZIP

Bab 2

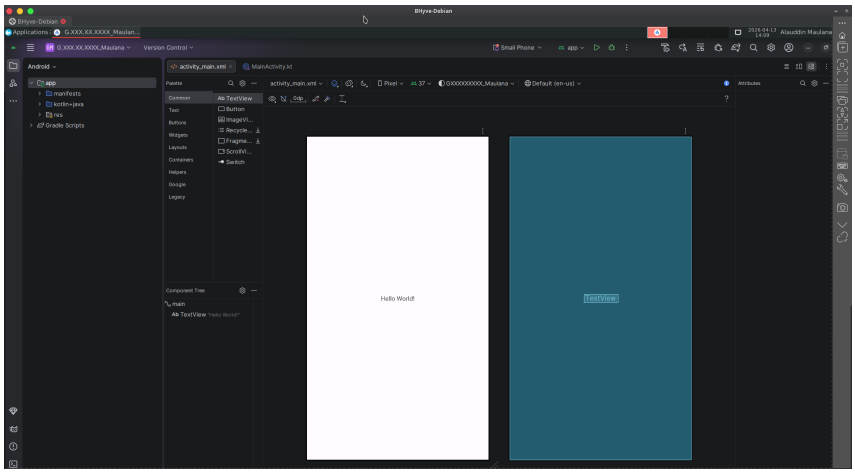
Struktur Proyek Android

2.1 Pendahuluan

Struktur proyek Android dan Activity Lifecycle merupakan dua konsep fundamental dalam pengembangan aplikasi berbasis Android yang menentukan bagaimana aplikasi dibangun, dijalankan, dan dikelola selama siklus hidupnya. Pemahaman terhadap struktur direktori proyek, seperti pengelolaan file manifest, kode sumber, serta resource, sangat penting untuk memastikan pengembangan yang terorganisasi dan mudah dipelihara. Di sisi lain, Activity Lifecycle mengatur transisi status sebuah activity sejak dibuat hingga dihancurkan, yang berpengaruh langsung terhadap manajemen memori, performa, dan pengalaman pengguna. Oleh karena itu, penguasaan kedua aspek ini menjadi dasar penting sebelum melanjutkan ke pengembangan aplikasi yang lebih kompleks.

2.2 Tutorial

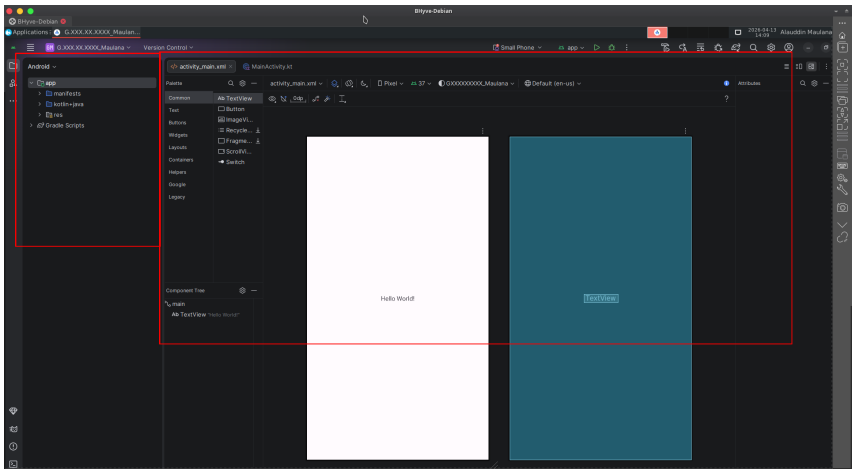
1. Buka *Android Studio*. Pastikan untuk membuka projek milik sendiri



Gambar 2.1: Membuka Projek Bab 1

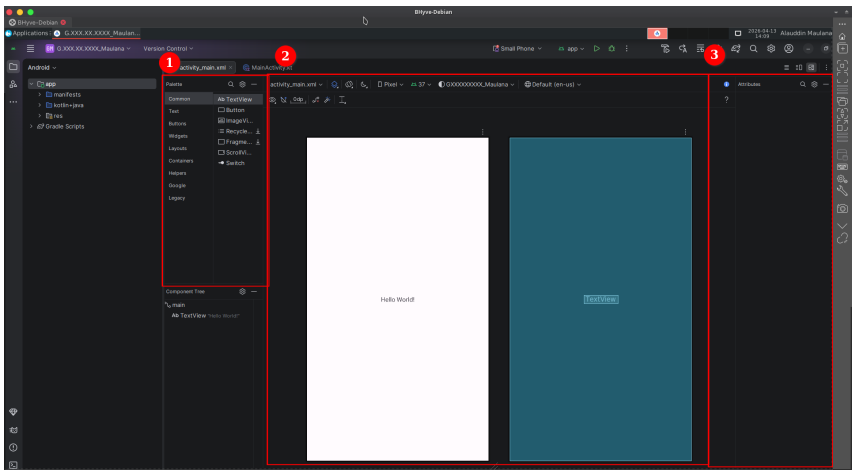
2. Jika membuka praktikum orang lain, klik **File**, lalu pilih **Close Project**. Lalu pilih projek praktikum yang sesuai. Jika tidak menemukan, ulangi Bab 1.

3. Di window Android Studio berikut, ada **2 Panel Utama** yang harus dikenali. Panel *Project* bagian kiri, dan Panel *Editor* bagian kanan.



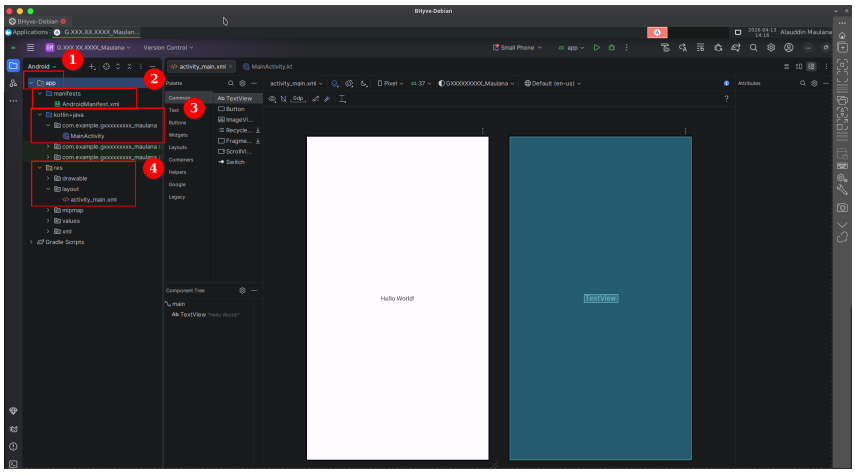
Gambar 2.2: Tampilan Utama

4. Bagian Panel *Editor* ada beberapa bagian penting: **Palette** (bagian kiri Editor), **Code / Design** (bagian tengah Editor), dan **Attribute** (panel kanan Editor Desainer)



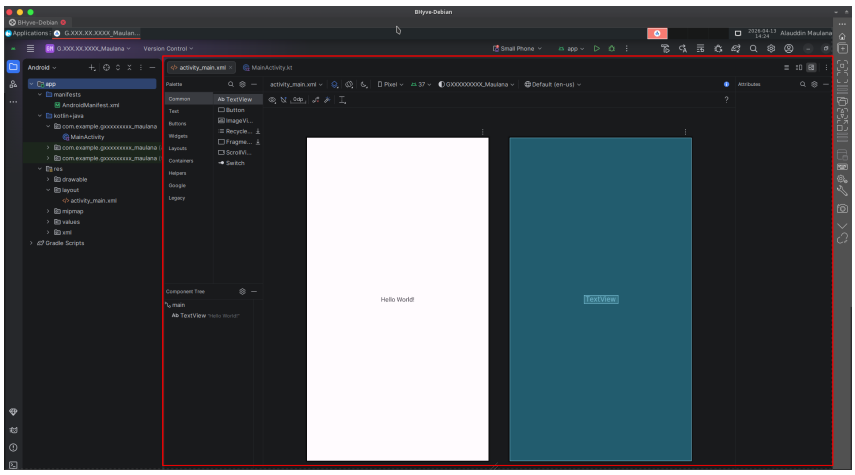
Gambar 2.3: Tampilan Desainer dan Kode

5. Alihkan fokus ke panel kiri **Project**. Perhatikan folder **app**. Folder ini berisikan:
   - folder *app* (utama)
   - folder *manifest* untuk mengendalikan perizinan dll
   - folder *kotlin+java* untuk fungsional Kotlin
   - folder *res* pendukung, khususnya folder *layout*

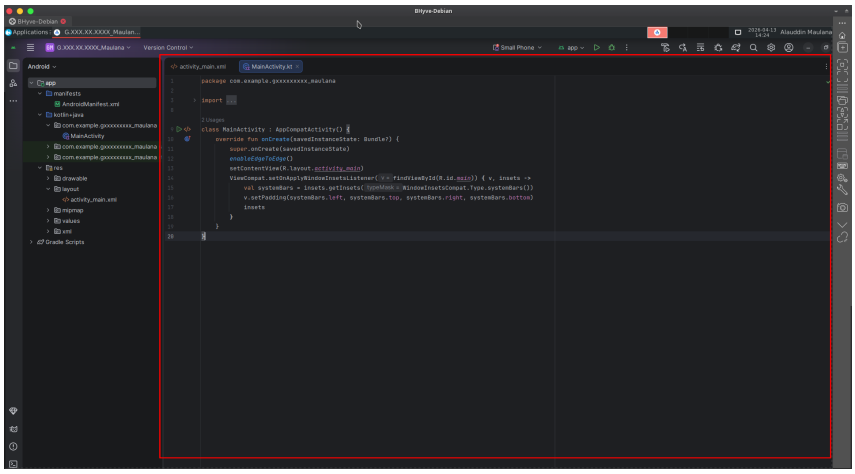


Gambar 2.4: Tampilan Panel Project

6. Di bagian **Editor**, tampilan bagian ini bergantung dari jenis file yang diedit. Jika **xml**, maka mode **UI Designer** dan jika **kt** akan memunculkan kode saja



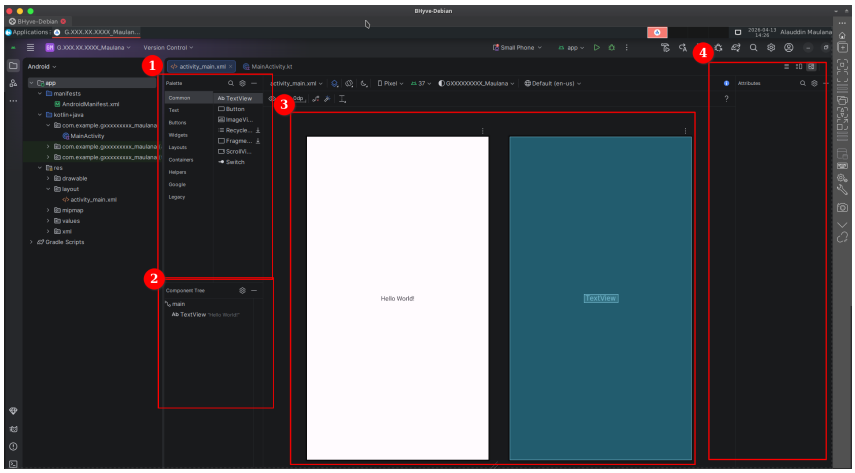
Gambar 2.5: Tampilan UI Designer XML



Gambar 2.6: Tampilan Kotlin Editor

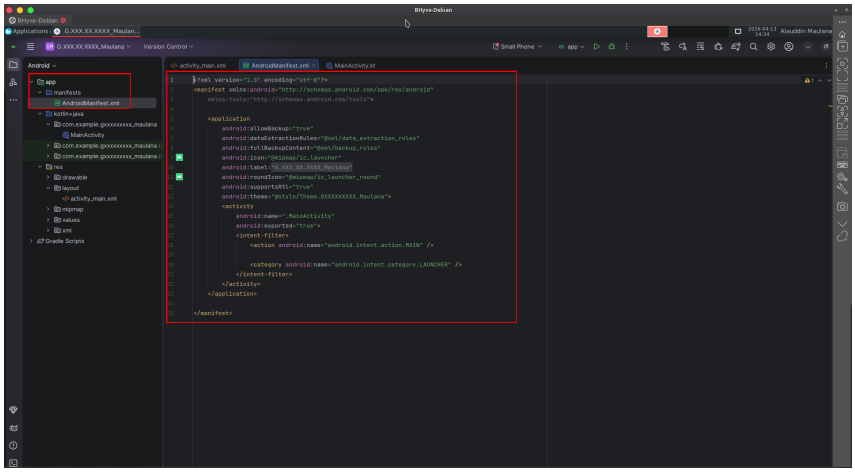
7. Di bagian **UI Designer**, terdapat 4 bagian terpenting:

- **Palette** : berisi widget Android
- **Component Tree** : daftar widget di layout
- **Layout Editor** : tempat membuat UI
- **Attribute** : modifikasi parameter angka, teks, dimensi



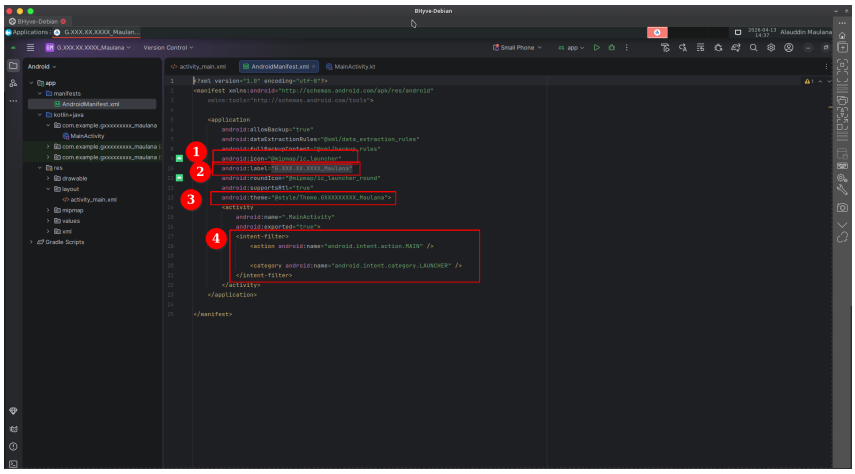
Gambar 2.7: Tampilan UI Desainer

8. Untuk mengenal lebih lanjut mengenai Android dan sistemnya. Buka terlebih dahulu file **AndroidManifest.xml** untuk memahami file ini.



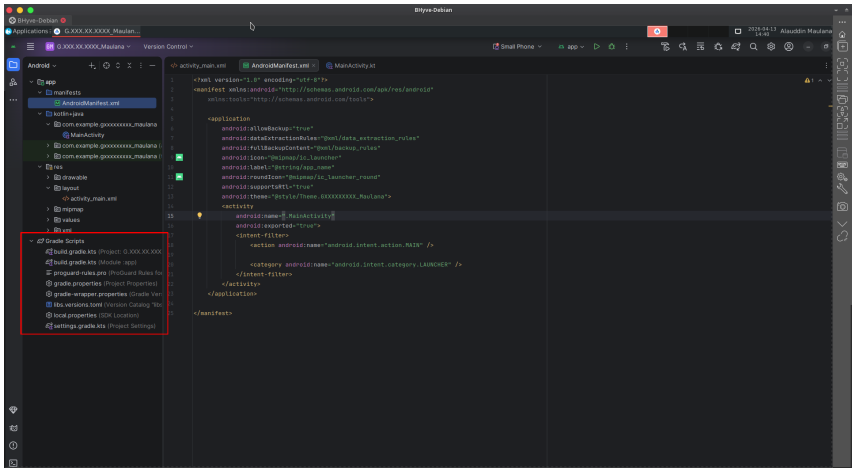
Gambar 2.8: File AndroidManifest.xml

9. File ini berfungsi untuk:
    - Mengatur Icon aplikasi setelah diinstall
    - Memberikan nama file
    - Mengatur tema
    - Mengatur halaman kepala



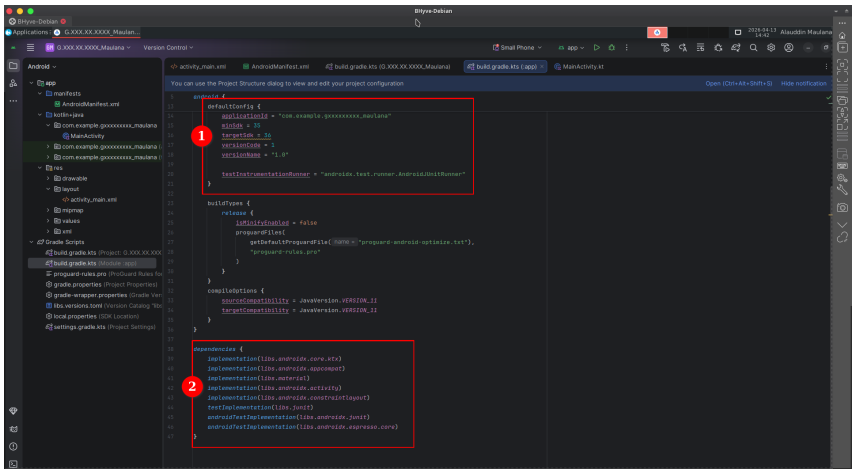
Gambar 2.9: Isi AndroidManifest.xml

10. Abaikan file **AndroidManifest.xml** dan jangan diubah. Cukup tutup saja.
11. Arahkan perhatian ke folder **Gradle Script** di panel **Project app** bagian kiri bawah.



Gambar 2.10: Folder Gradle Script

12. File ini tidak boleh diubah kecuali kondisi tertentu saja. Fungsi kode ini:
    - Mengatur minimum SDK
    - Mengatur ketergantungan (library lain)



Gambar 2.11: Isi build.gradle.kts (Module app)

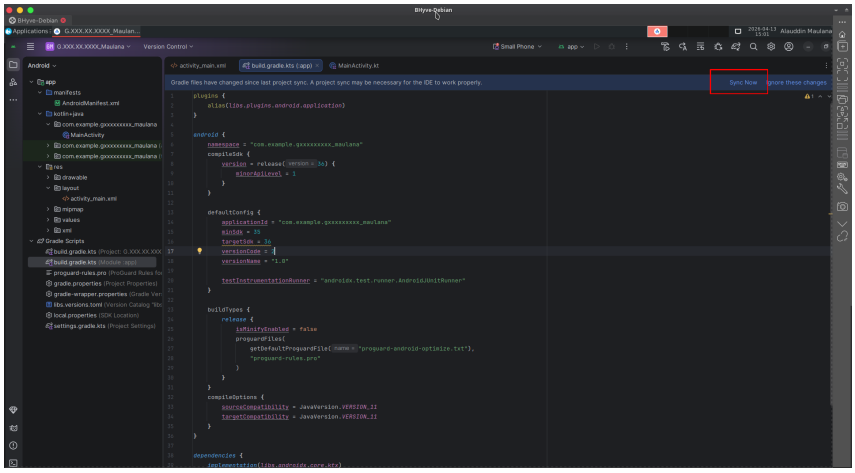
13. Bagian yang tidak kalah penting adalah proses **Build** dalam Android Studio. Untuk bisa berjalan dengan baik, Android Studio harus menyelesaikan beberapa tahapan:

    (a) **Gradle Sync** : Proses sangat penting untuk memastikan semua dependensi dan konfigurasi lengkap. Tanpa ini, Android Studio akan gagal. Hanya berjalan sekali kecuali ada perubahan Gradle

    (b) **Build Project** : Proses ini membangun kode menjadi APK

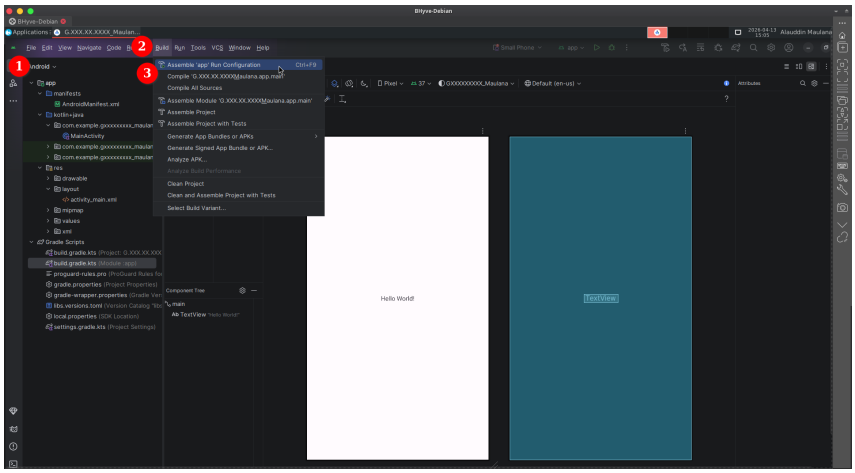
    (c) **Run App** : Menjalankan APK ke Emulator

14. **Gradle Sync** hanya bisa dilakukan jika **build.gradle.kts** mengalami perubahan, dan membutuhkan download



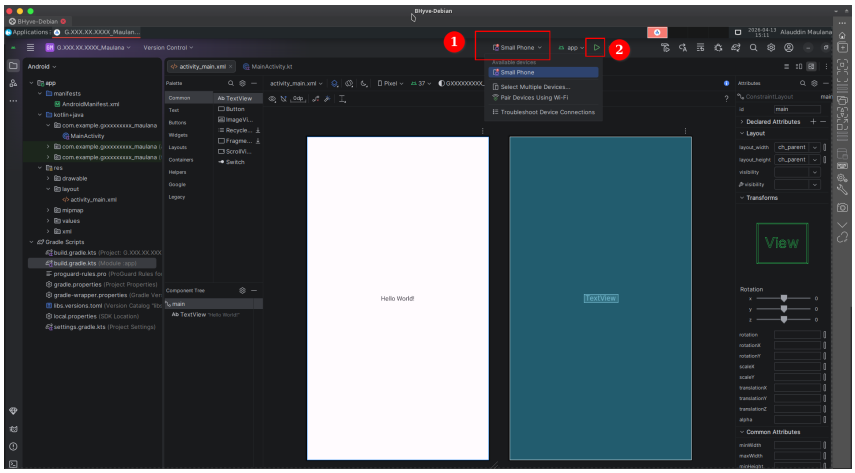
Gambar 2.12: Contoh **Sync Now** ketika **build.gradle.kts** dirubah

15. Sedangkan **Build Project** dapat dilakukan secara manual selama **Gradle Sync** sukses. Untuk melakukan **Build** manual, klik **Empat Strip**, pilih menu **Build**, dan pilih **Assemble app**



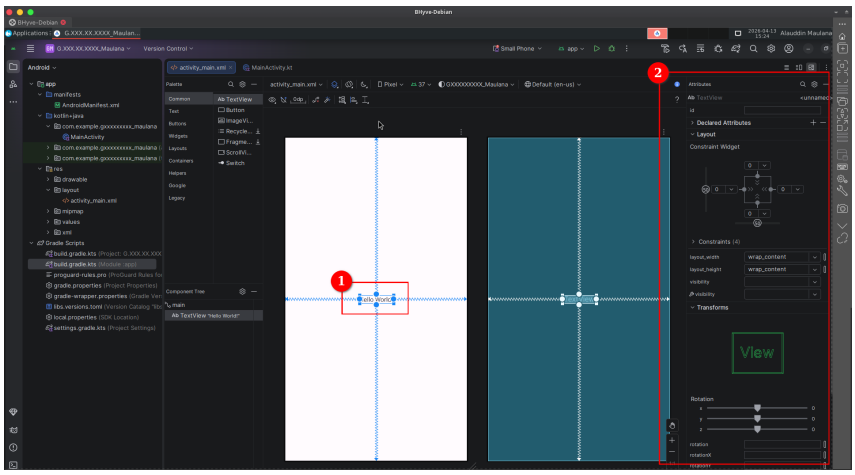
Gambar 2.13: Manual Building

16. Untuk **Run App**, dapat dilakukan dengan mudah. Cukup pilih perangkat target (Emulator / Perangkat Asli), lalu klik **Tombol Play**



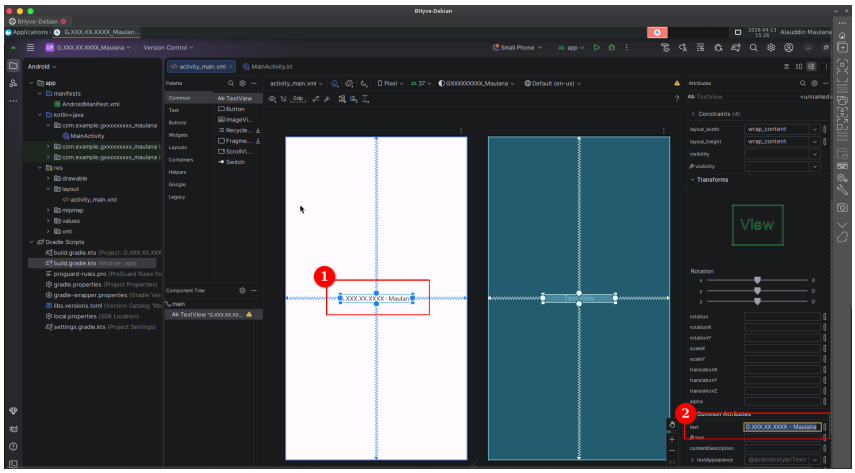
Gambar 2.14: Run App

17. Berikutnya adalah **Attribute**. Bagian ini digunakan untuk mengubah komponen pada widget. Sebagai contoh, pada mode UI Desainer klik tulisan **Hello World** sehingga panel kanan **Attribute** akan aktif.



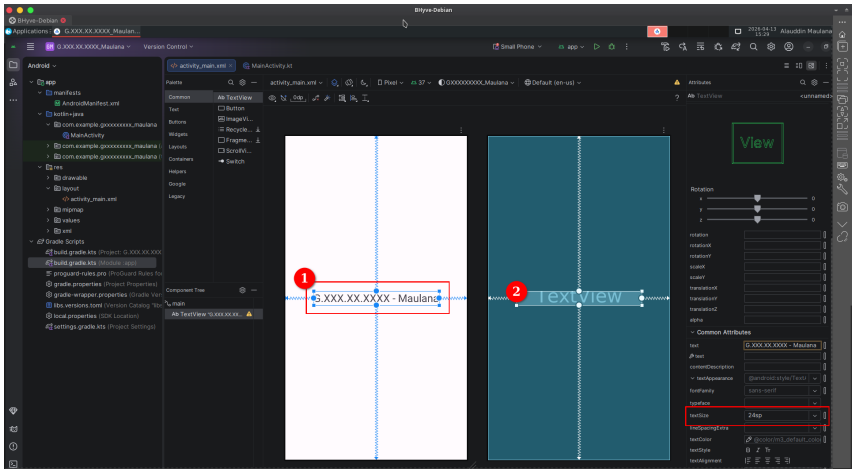
Gambar 2.15: Menggunakan Panel Attribute

18. Cari attribute **Text**, dan ubah menjadi NIM - Nama.



Gambar 2.16: Ubah Text Attribute

19. Dengan menggunakan attribute **textSize** ubah ukuran font menjadi 24sp



Gambar 2.17: Mengubah Ukuran Teks

20. Setelah selesai, **Export to ZIP** dan kirim ke e-learning

Bab 3

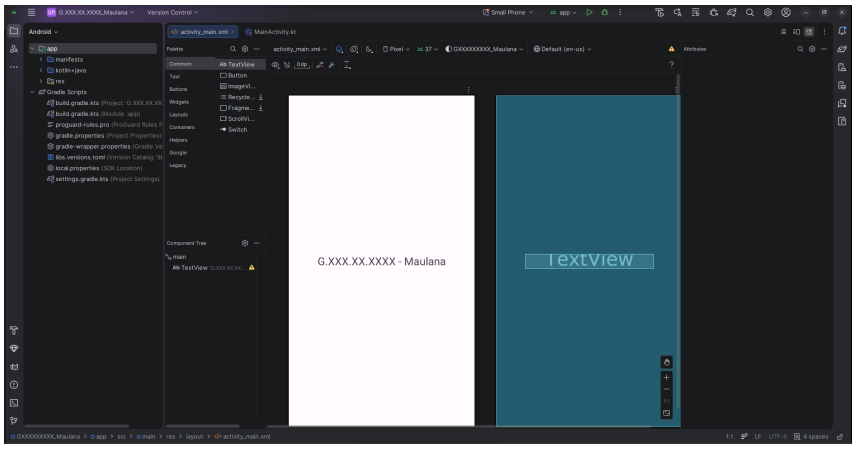
Desain Antarmuka XML

3.1 Pendahuluan

Praktikum ini bertujuan untuk memberikan pemahaman dasar mengenai perancangan antarmuka pengguna (User Interface/UI) pada aplikasi Android menggunakan pendekatan deklaratif berbasis XML. Fokus utama kegiatan adalah pemanfaatan ConstraintLayout sebagai layout modern yang fleksibel serta implementasi komponen UI dasar seperti TextView, EditText, dan Button untuk membangun tampilan form sederhana yang terstruktur dan responsif. Melalui langkah-langkah praktis, mahasiswa diharapkan mampu memahami konsep penempatan komponen berbasis constraint, pengaturan atribut visual, serta prinsip pemisahan antara desain antarmuka dan logika program, yang merupakan praktik standar dalam pengembangan aplikasi mobile modern.

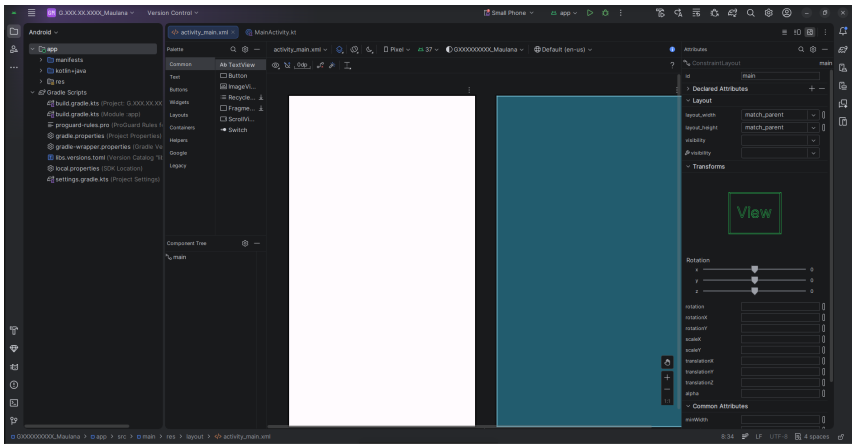
3.2 Tutorial

1. Buka kembali Android Studio, dan pastikan untuk membuka projek mahasiswa masing-masing.



Gambar 3.1: Membuka Android Studio

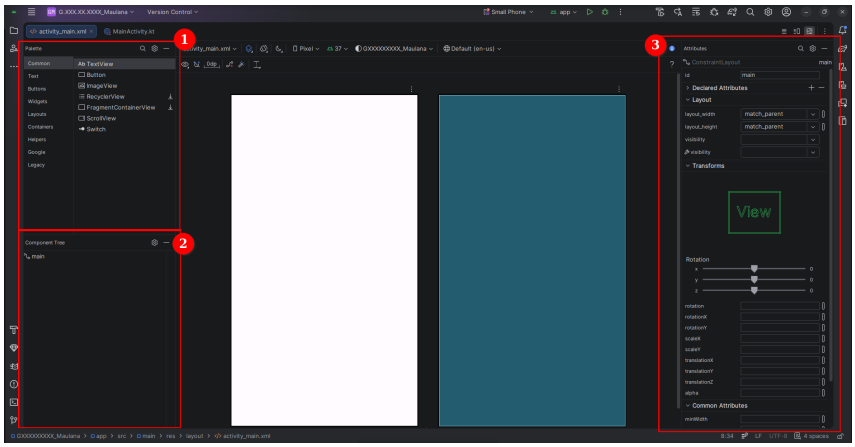
2. Hapus **TextView** yang ada dan meninggalkan kanvas kosong



Gambar 3.2: Menghapus TextView

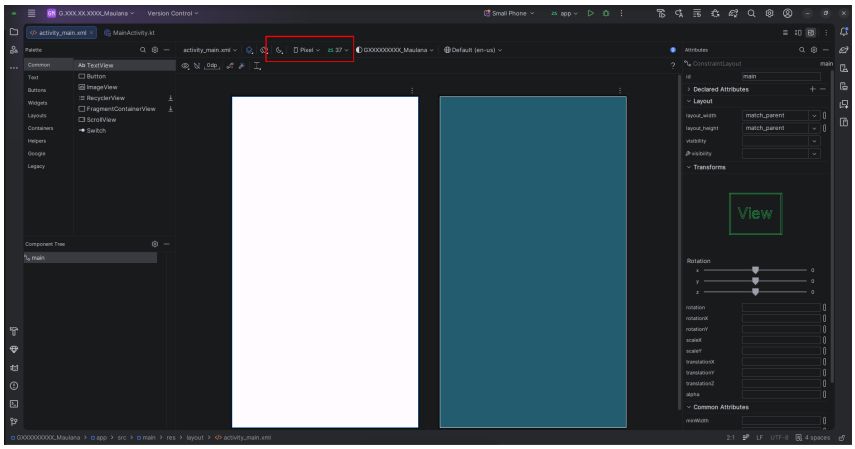
3. Mengenal **Pallette**, **Component Tree**, dan **Attribute**. Bagian ini memiliki fungsi:

   - **Palette** : daftar widget yang dapat digunakan untuk layout Android
   - **Component Tree** : daftar widget yang digunakan dan sudah diletakkan di dalam layout
   - **Attribute** : panel untuk memodifikasi sifat atau tampilan widget

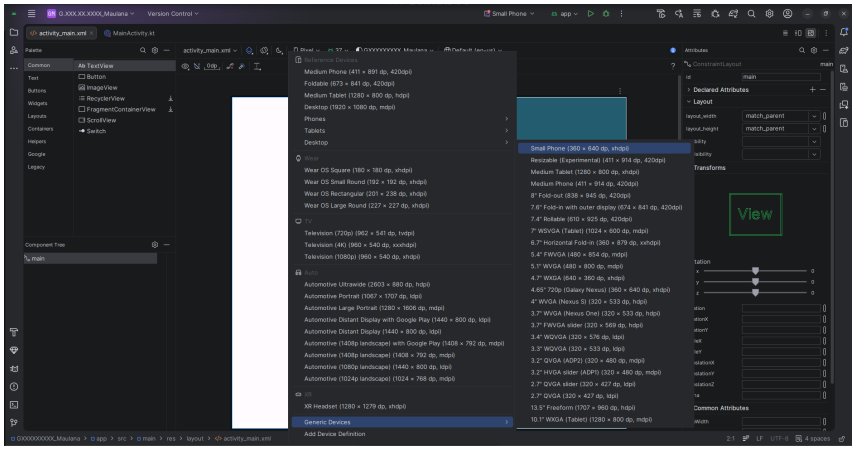


Gambar 3.3: Komponen Penting UI Desainer

4. Berikutnya adalah menyesuaikan ukuran layout dengan perangkat / emulator. Cek bagian atas layout, di sana terdapat beberapa icon. Klik icon dengan tulisan **Pixel** disebelah icon **bulan sabit**. Ubah menjadi **Small Phone** dengan cara : **Generic Devices** → **Small Phone**

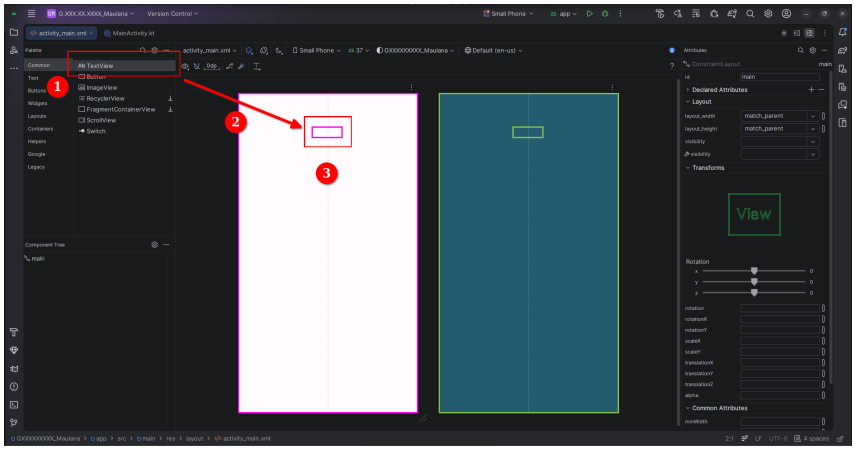


Gambar 3.4: Konfigurasi ke Small Phone

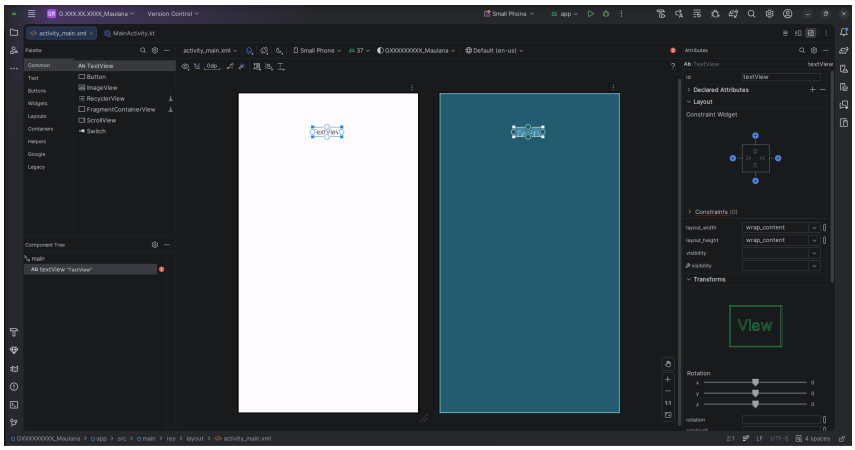


Gambar 3.5: Konfigurasi ke Small Phone

5. Setelah Android Studio menyesuaikan layout ke **Small Phone**, maka langkah berikutnya adalah menambahkan widget ke layar. Klik-dan-Drag **TextView** ke Layout

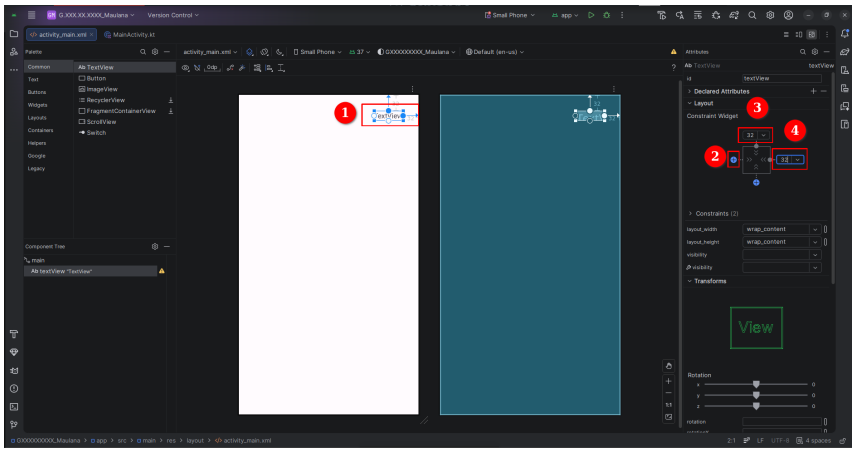


Gambar 3.6: Menarik TextWdget ke Layout



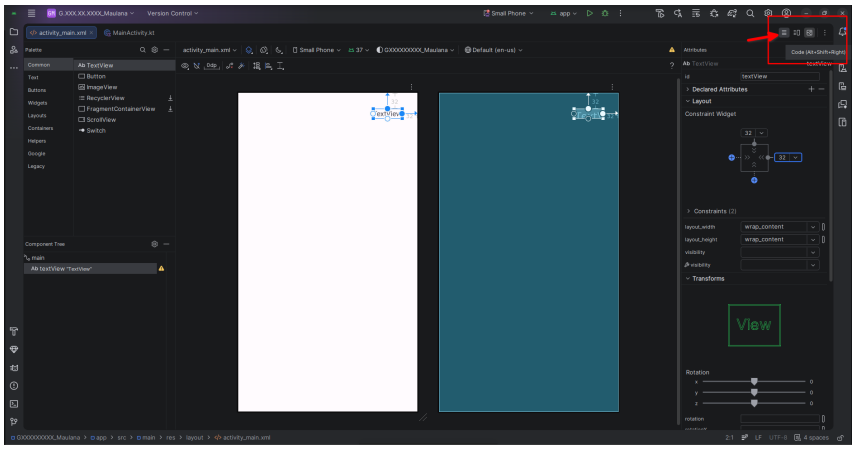
Gambar 3.7: Hasil Layout

6. Di sisi kanan dengan keadaan **TextView** aktif terklik, atur **Contraint** untuk **Kanan, Kiri,** dan **Atas** dengan cara klik **Plus** + dan atur margin menjadi **32dp** (tanpa tulis dp)

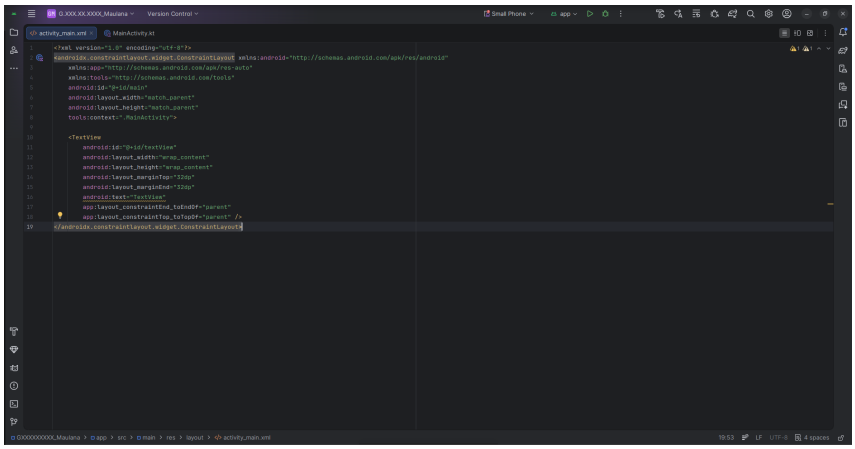


Gambar 3.8: Mengkonfigurasikan Atribut Contraint TextWidget

7. Untuk melihat kode sumber dalam format XML, dapat dilakukan dengan mengklik tombol **Code** di sebelah **Kanan-Atas**

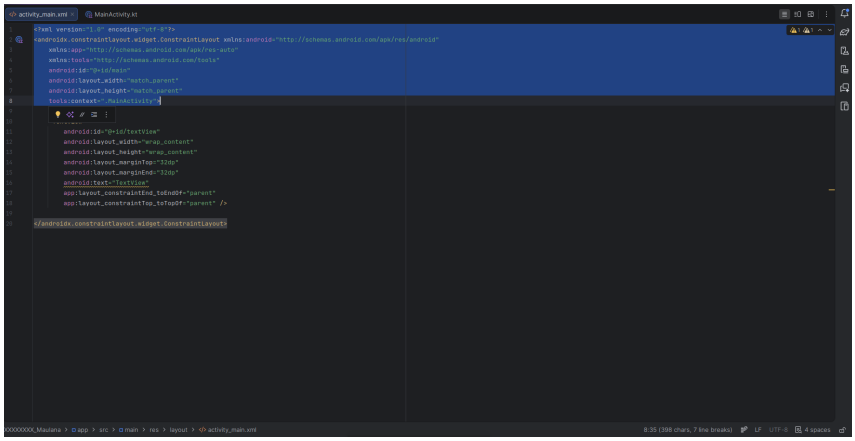


Gambar 3.9: Pindah Mode Code

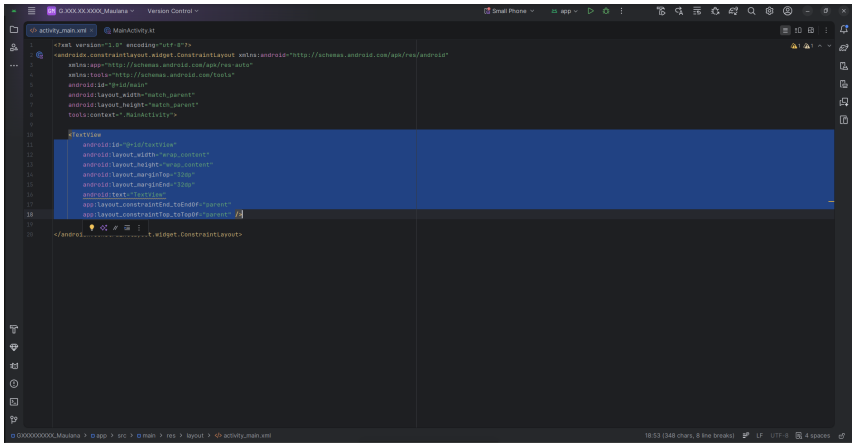


Gambar 3.10: Mode Tampilan Kode

8. Kode ini terdapat dua bagian berupa: **Kode Layout**, dan **Kode Widget**.
    - **Kode Layout** : Membungkus kode widget dengan konfigurasi tampilan layout
    - **Kode Widget** : Terdapat di dalam kode widget

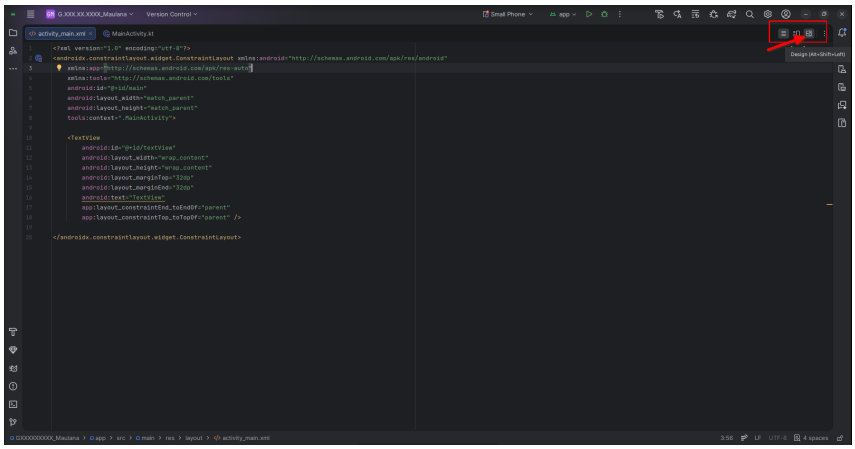


Gambar 3.11: Kode Layout



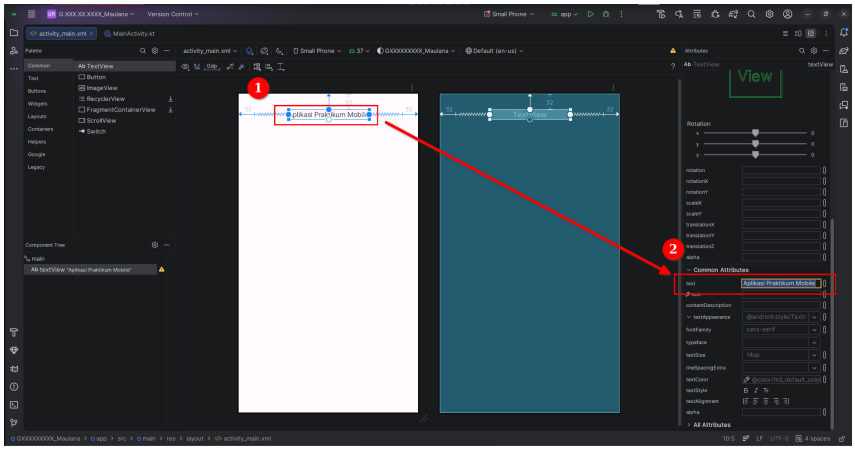
Gambar 3.12: Kode Widget

9. Untuk kembali ke mode UI Desainer, klik kembali bagian **Kanan-Atas**



Gambar 3.13: Kembali Mode UI Desainer

10. Klik **TextView** yang telah dibuat tadi. Ubah **Attribute** berikut ini:
    - **Text** : Aplikasi Praktikum Mobile

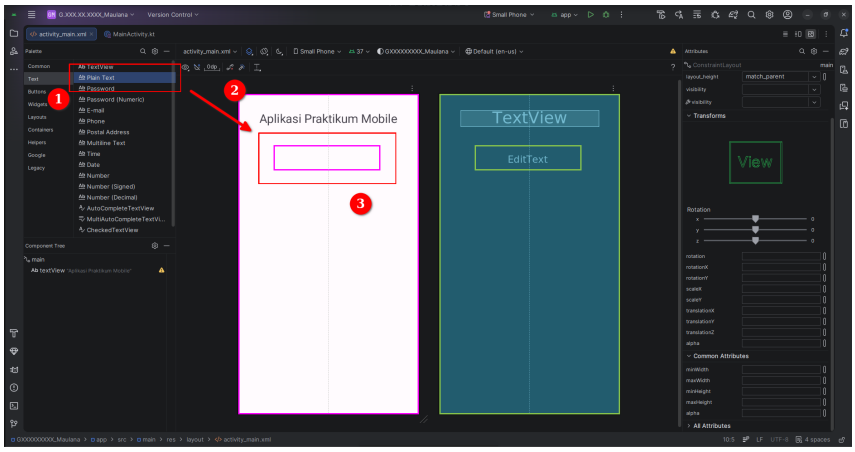


Gambar 3.14: Ubah Atribut Text

11. Selain itu ubah **Attribut** berikut ini:
    - **TextSize** : 24sp
    - **TextStyle** : Bold

Gambar 3.15: Ubah Atribut TextSize dan TextStyle

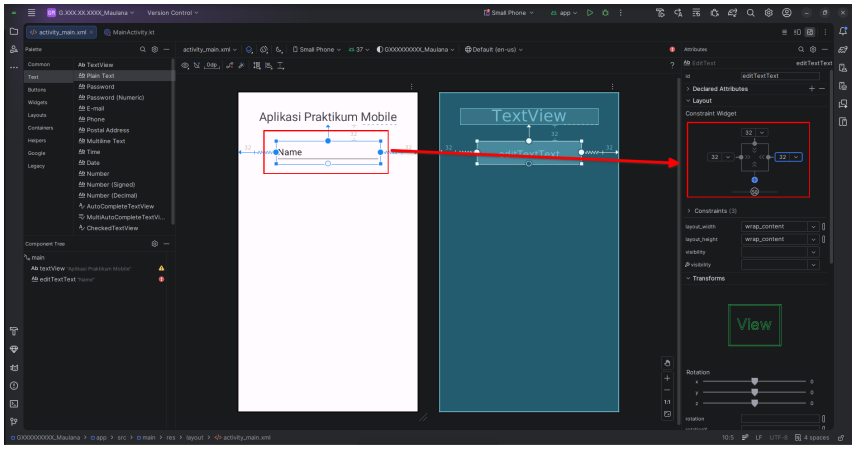
12. Berikutnya tambahkan **1 Wdiget** baru berupa **EditText: Plain Text** di bawah **TextView**



Gambar 3.16: Menambahkan EditText

13. Sama seperti **TextView** atur **Constraint** terlebih dahulu dengan konfigurasi:

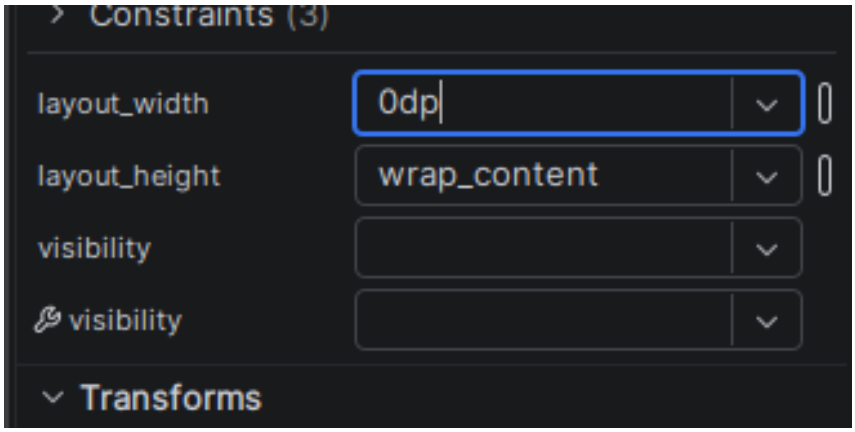
- **Top** : 32dp
- **Left** : 32dp
- **Right** : 32dp



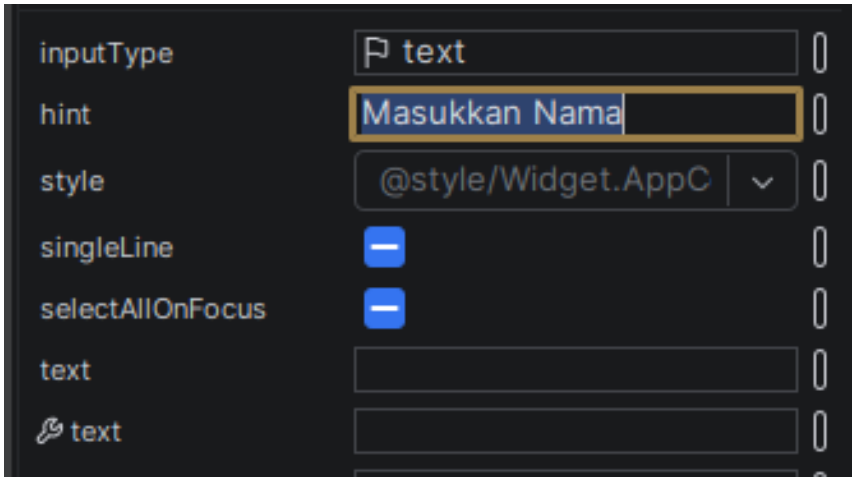
Gambar 3.17: Konfigurasi Contraint EditText

14. Lalu konfigurasikan **Attribute** dari **EditText** seperti berikut:

- **Text** : (kosong)
- **Hint** : Masukkan Nama
- **layout_width** : 0dp (match_constraint)



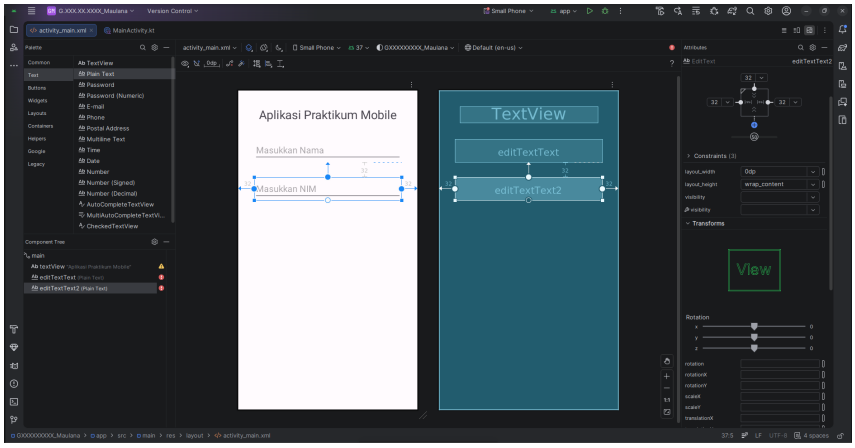
Gambar 3.18: Konfigurasi layout_width



Gambar 3.19: Konfigurasi Text dan Hint

15. Tambahkan **EditText: Plain Text** baru di bawah **EditText** untuk Nama, namun kali ini **Attribute** diganti dengan

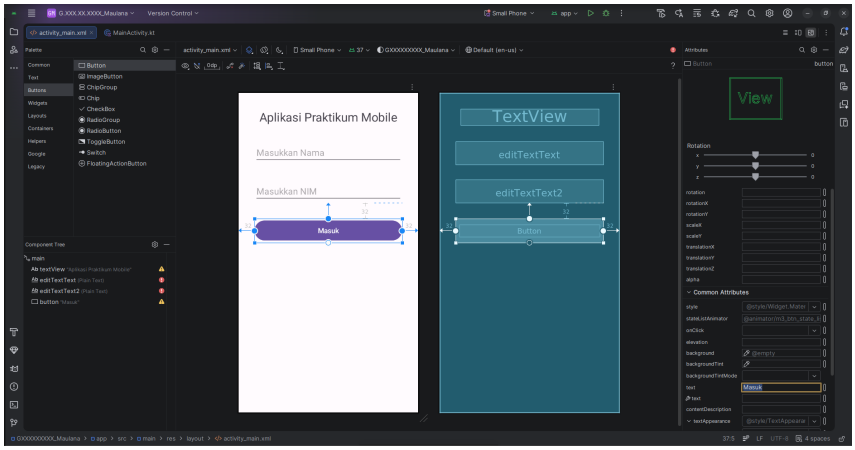
    - **Text** : (kosong)
    - **Hint** : Masukkan NIM
    - **layout_width** : 0dp (match_constraint)



Gambar 3.20: EditText NIM

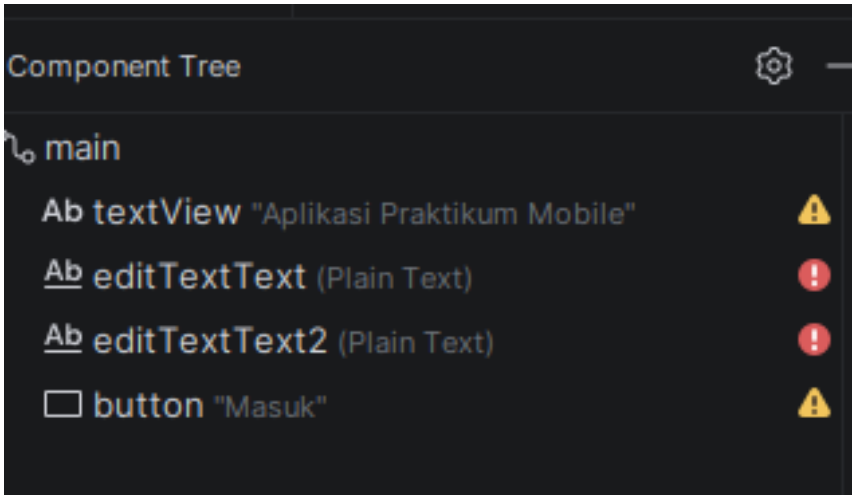
16. Widget yang terakhir ditambahkan adalah **Button**. Klik-dan-Drag **Button** ke Layout. Atur sama persis dengan **EditText** dengan konfigurasi

    - **Text** : Masuk
    - **layout_width** : 0dp (match_constraint)



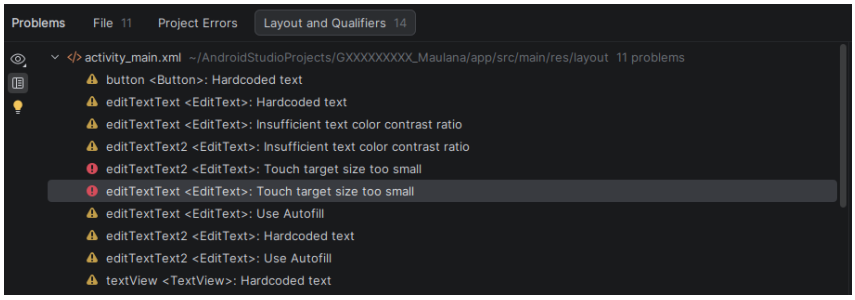
Gambar 3.21: Menambahkan Button

17. Setelah semua widget masuk ke layout, langkah berikutnya adalah menghilangkan **Error** di **Component Tree**



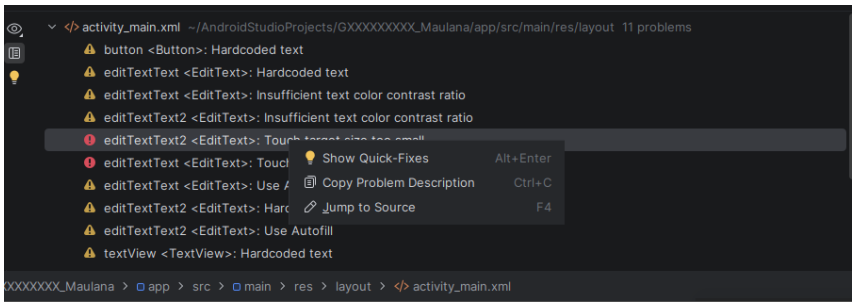
Gambar 3.22: Component Tree

18. **Klik** tanda **Error** di **EditText**, maka akan muncul panel baru di bawah.

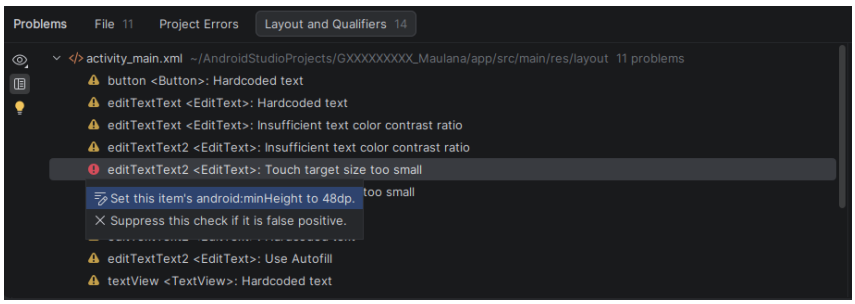


Gambar 3.23: Panel Problems

19. Untuk menyelesaikan **Error: Touch Target Too Small**, Klik Kanan → pilih **Show Quick-Fixes** → pilih **Set this item's android::minHeight to 48dp.**



Gambar 3.24: Panel Problems



Gambar 3.25: Panel Problems

20. Ulangi langkah di atas untuk menghilang semua **Error**.
21. Verifikasi projek dengan menjalankan ke Emulator / Perangkat.
22. Ekspor ke ZIP dan kirim ke e-Learning

Bab 4

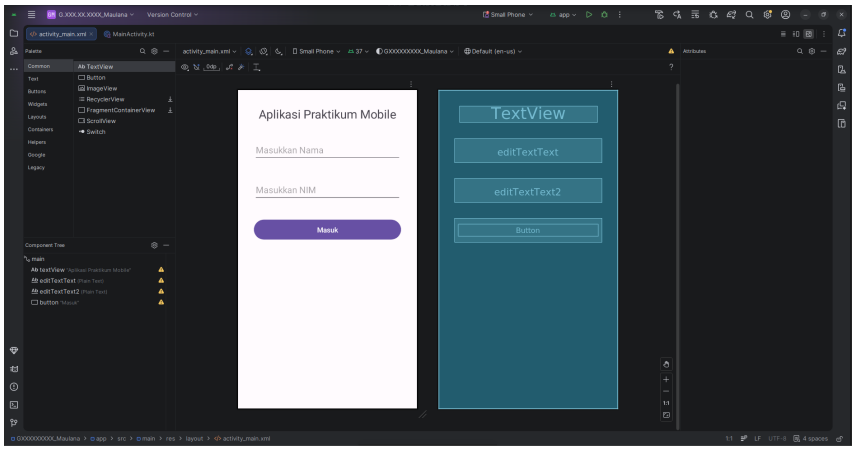
Dasar Kotlin dalam Android

4.1 Pendahuluan

Praktikum Dasar Kotlin dalam Android bertujuan memberikan pemahaman fundamental mengenai penggunaan bahasa pemrograman Kotlin dalam pengembangan aplikasi Android, mulai dari pengenalan sintaks dasar, struktur proyek, hingga implementasi interaksi antara antarmuka pengguna (UI) dan logika program. Kegiatan praktikum ini dirancang secara bertahap melalui pendekatan hands-on agar mahasiswa mampu memahami konsep variabel, fungsi, kontrol alur, serta event handling dalam konteks aplikasi nyata. Selain itu, praktikum ini juga menekankan kemampuan dalam mengelola input pengguna, melakukan debugging sederhana, serta membangun aplikasi interaktif berbasis Android Studio. Dengan mengikuti seluruh rangkaian praktikum, diharapkan mahasiswa memiliki dasar yang kuat untuk mengembangkan aplikasi Android yang lebih kompleks dan terintegrasi dengan berbagai teknologi lanjutan.

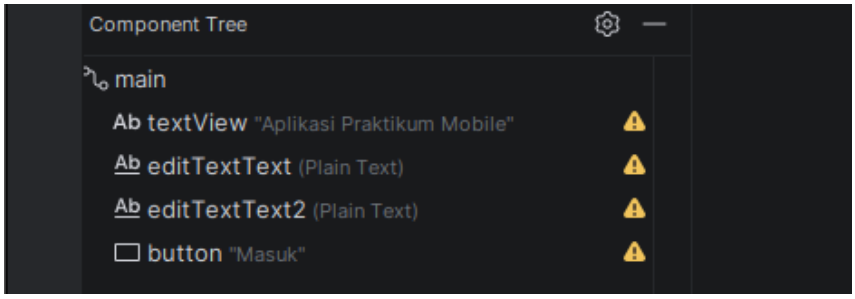
4.2 Tutorial

1. Buka kembali projek yang telah dibuat sebelumnya, dan pastikan bahwa antarmuka telah ready dari Bab 3



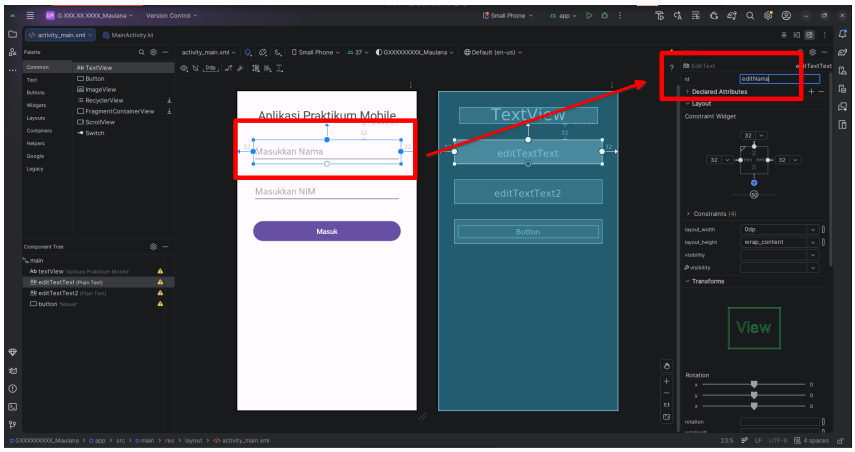
Gambar 4.1: Membuka Praktikum

2. Untuk memulai pemrograman menggunakan Kotlin di Android, langkah pertama yang harus dilakukan adalah meregistrasikan objek-objek yang ada di activity_main.xml
3. Cek bagian **Component Tree**



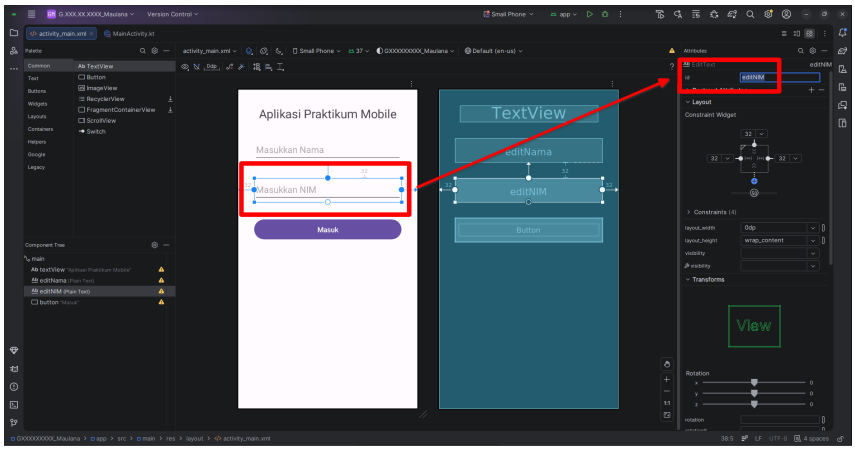
Gambar 4.2: Cek Komponen Tree

4. Ubah **ID Attribute** dari objek **EditText Nama**. Klik Objek tersebut dan ubah **ID** menjadi **editNama**



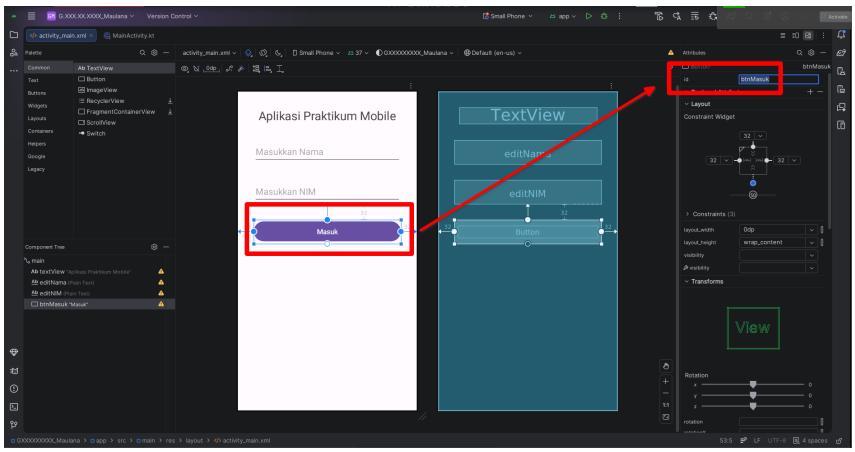
Gambar 4.3: Mengubah ID editNama

5. Tekan **Enter**, dan klik **Refactor**
6. Klik objek kedua di bawah nya, lalu ubah **ID** menjadi **editNIM**



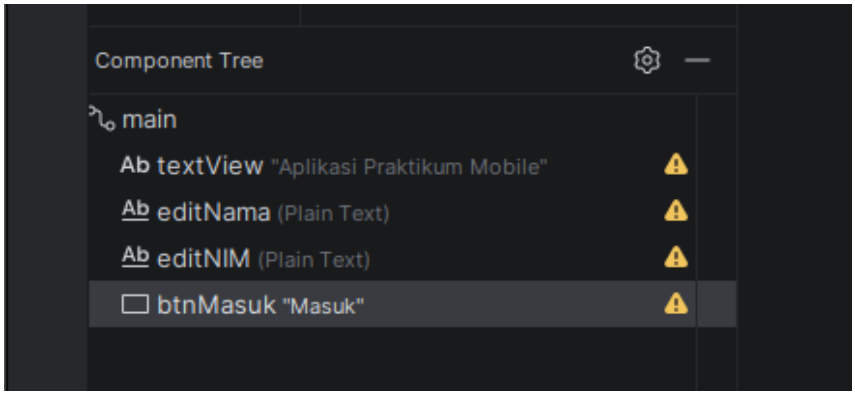
Gambar 4.4: Mengubah ID editNIM

7. Tekan **Enter** lalu klik **Refactor**
8. Klik **Button** di bawahnya dan ubat menjadi **btnMasuk**



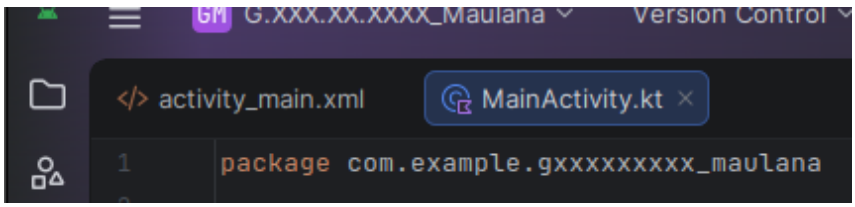
Gambar 4.5: Mengubah ID btnMasuk

9. Verifikasi ulang dengan mengecek **Component Tree**, pastikan Layout **main** tidak berubah



Gambar 4.6: Cek Ulang Component Tree

10. Untuk memulai pemrograman dengan menggunakan Kotlin setelah mengkonfigurasikan ID, klik **MainActivity.ky**



Gambar 4.7: Membuka File MainActivity.kt

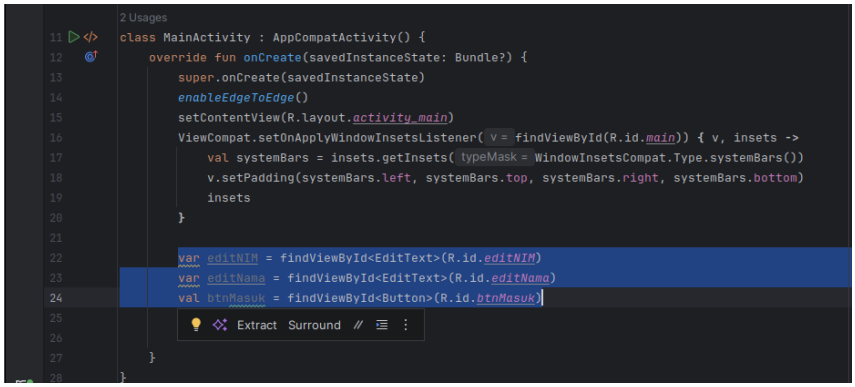
11. Cek kode di dalam **MainActivity.kt**, cek simbol } dan sisakan 2 simbol di bawah. Cek gambar

Gambar 4.8: Memulai Pemrograman

12. Untuk menginisialisasi benda-benda yang ada di layout, masukkan kode berikut tepat ditunjukkan oleh garis

Potongan Kode

```
var editNIM = findViewById<EditText>(R.id.editNIM)
var editNama = findViewById<EditText>(R.id.editNama)
val btnMasuk = findViewById<Button>(R.id.btnMasuk)
```

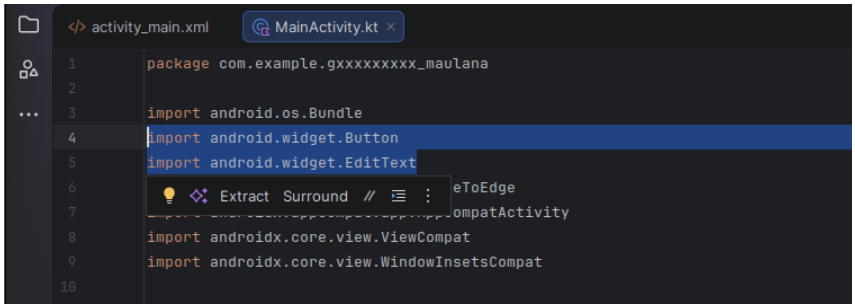


Gambar 4.9: Memasukkan Kode Inisialisasi Objek

13. Kode yang dimasukkan akan berwarna merah, untuk menyelesaikan masalah ini, masukkan kode berikut di bagian atas

Potongan Kode

```
import android.widget.Button
import android.widget.EditText
```

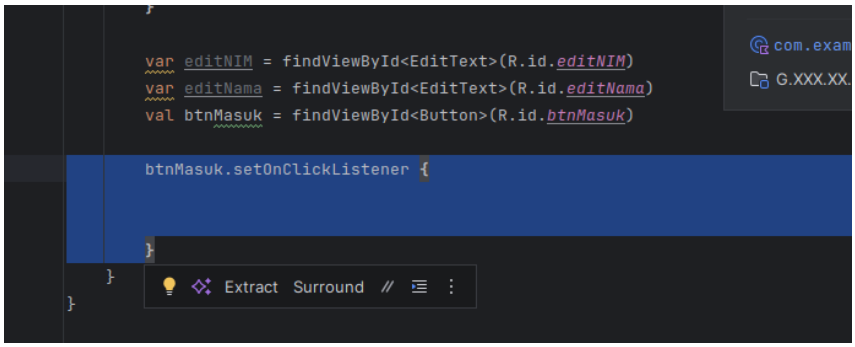


Gambar 4.10: Impor Library EditText dan Button

14. Kode ini belum mengaktifkan interaksi di halaman depan. Masukkan kode berikut setelah inisialisasi objek yang sudah dibuat sebelumnya

Potongan Kode

```
btnMasuk.setOnClickListener {

}
```

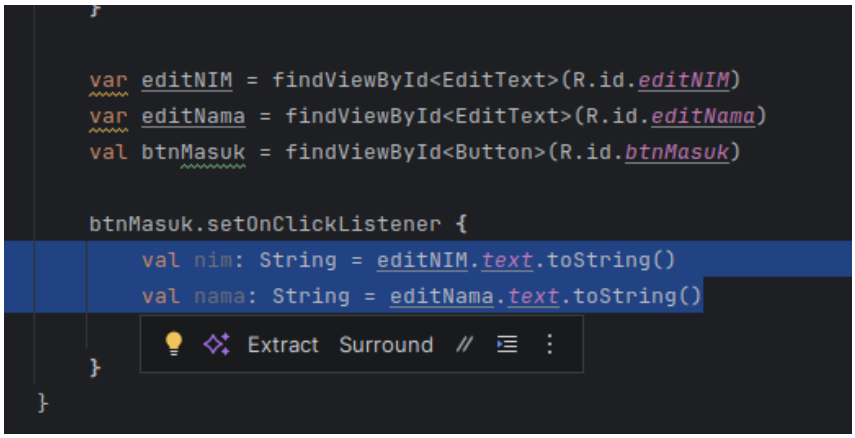


Gambar 4.11: Menambahkan Listener ke Button Masuk

15. Agar dapat menarik data teks yang diinputkan di halaman depan, masukkan kode berikut di dalam listener button

Potongan Kode

```
val nim: String = editNIM.text.toString()
val nama: String = editNama.text.toString()
```

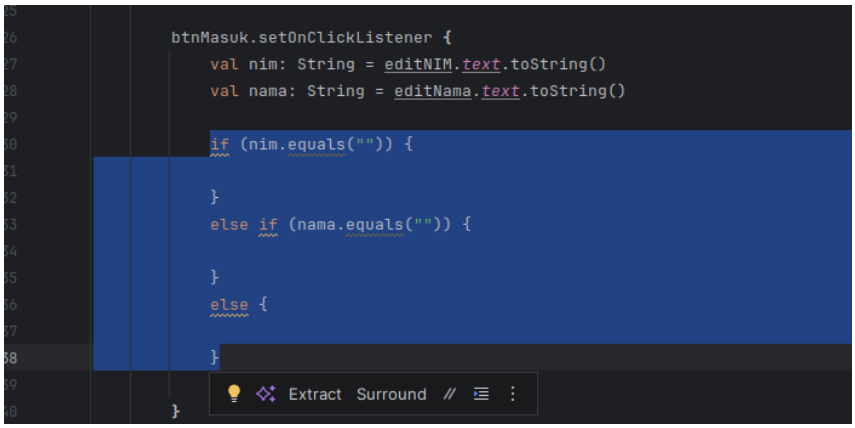


Gambar 4.12: Menambahkan Kode Mengambil Data

16. Data yang diambil tadi kemudian ditampilkan dengan menggunakan **Toast**. Perhatikan kode berikut untuk mengecek apakah kosong atau tidak

Potongan Kode

```
if (nim.equals("")) {

}
else if (nama.equals("")) {

}
else {

}
```

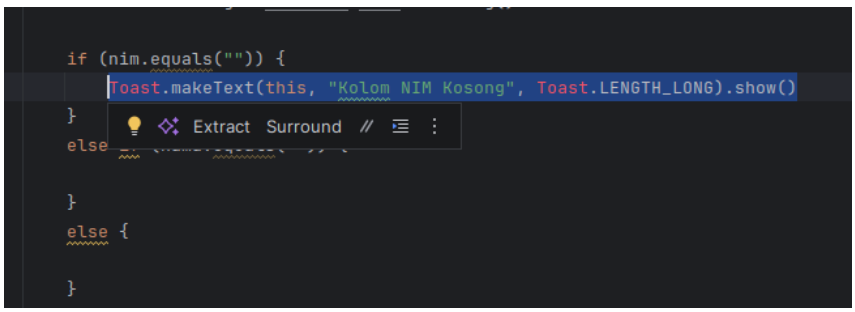


Gambar 4.13: Mengecek Data Teks

17. Masukkan kode berikut di blok **IF** terlebih dahulu

Potongan Kode

```
Toast.makeText(this, "Kolom NIM Kosong", Toast.LENGTH_LONG).show()
```

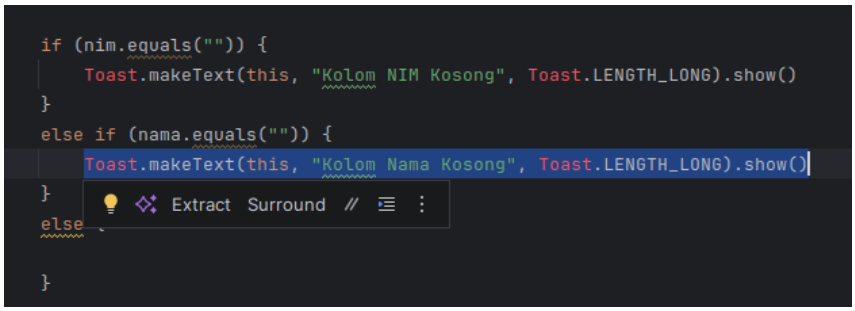


Gambar 4.14: Kode Aksi Jika NIM Kosong

18. Lanjutkan kode untuk blok **ELSE IF**

Potongan Kode

```
Toast.makeText(this, "Kolom NIM Kosong", Toast.LENGTH_LONG).show()
```

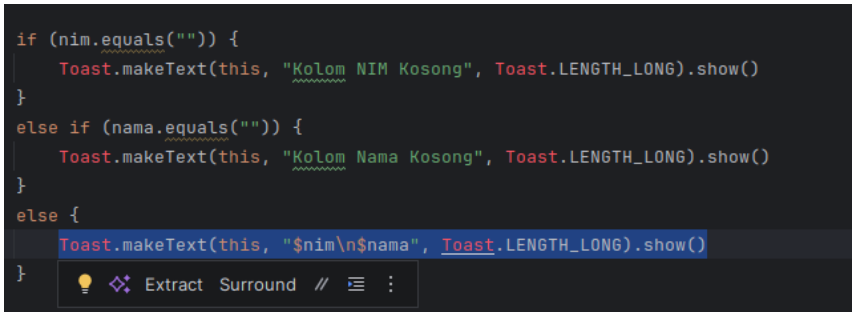


Gambar 4.15: Kode Aksi Jika Nama Kosong

19. Dan dibagian blok **ELSE**, masukkan kode berikut

Potongan Kode

```
Toast.makeText(this, "$nim\n$nama", Toast.LENGTH_LONG).show()
```

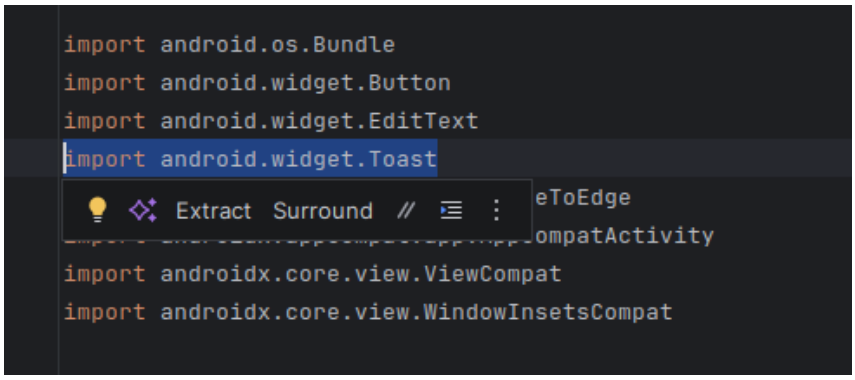


Gambar 4.16: Kode Aksi Sesudah Lengkap

20. Jika kode **Toast** masih berwarna merah, masukkan kode berikut di bagian atas

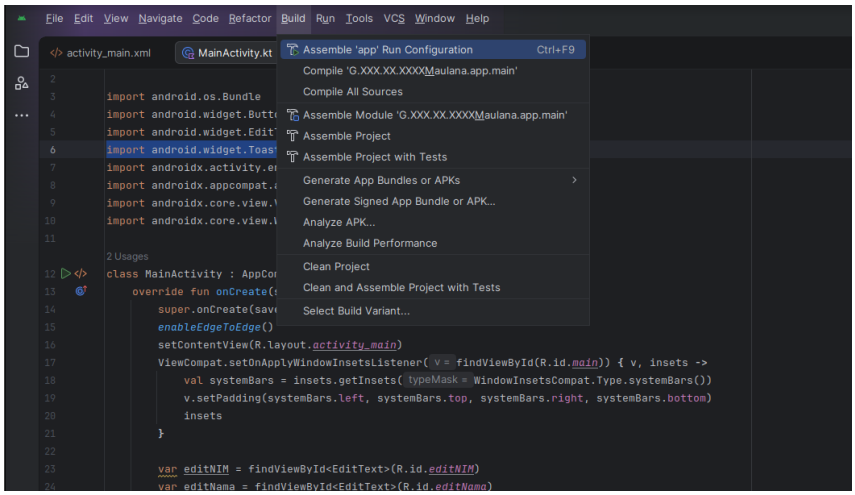
Potongan Kode

```
import android.widget.Toast
```

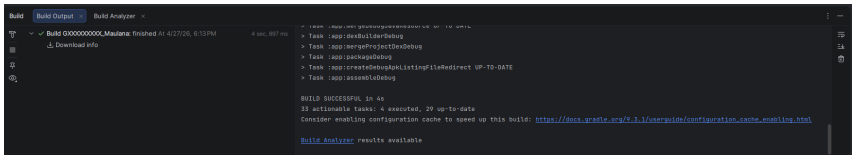


Gambar 4.17: Menambahkan Import Toast

21. Untuk menguji apakah kode sudah benar atau tidak, klik **Strip Empat** lalu menu **Build**, dan pilih **Assemble 'app' Configuration**

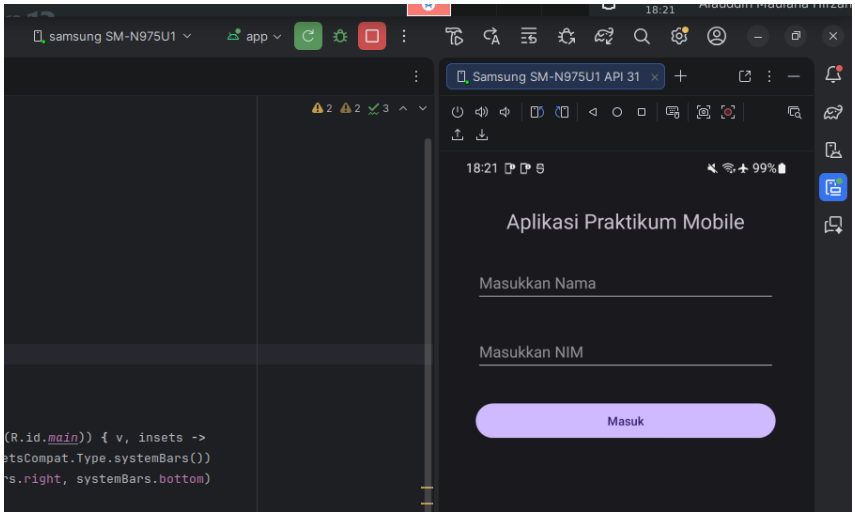


Gambar 4.18: Manual Building

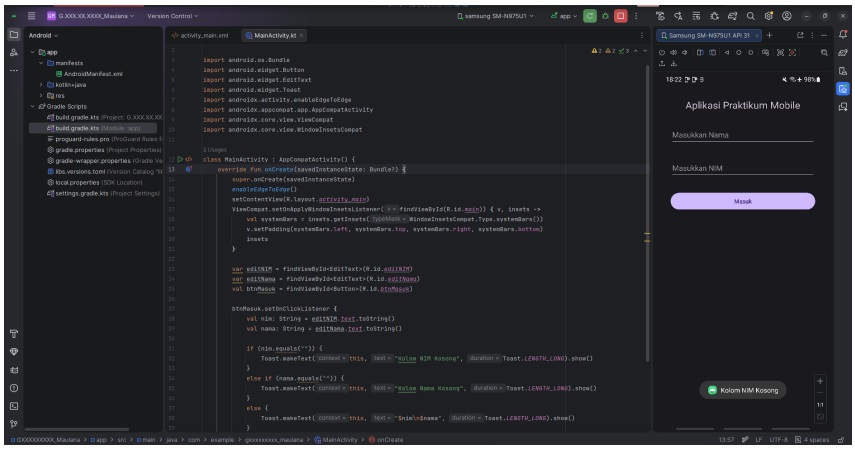


Gambar 4.19: Building Sukses

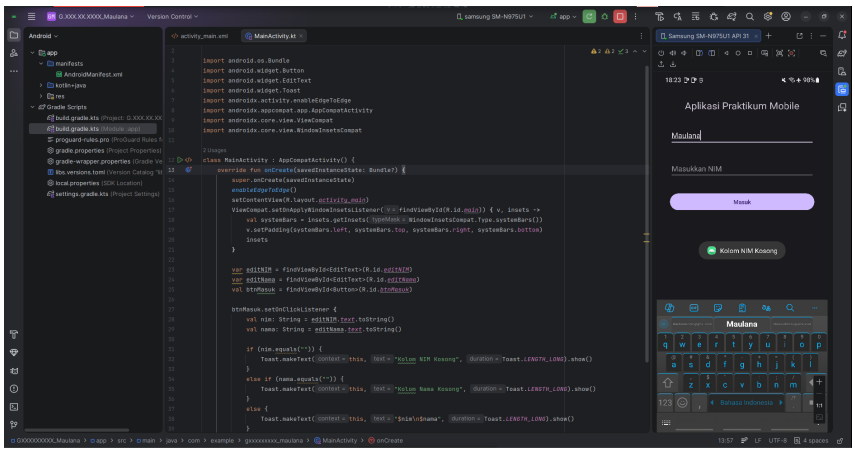
22. Jalankan aplikasi dengan klik **Play** dan lihat hasilnya



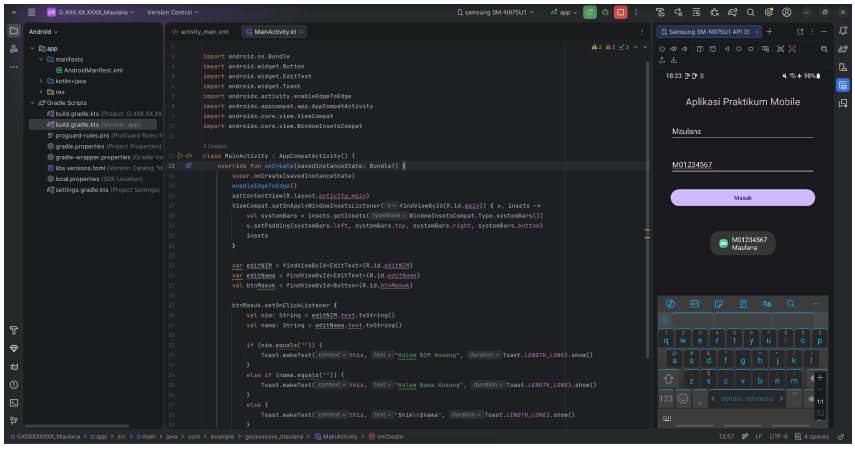
Gambar 4.20: Tampilan Aplikasi



Gambar 4.21: Tampilan Error Nama



Gambar 4.22: Tampilan Error NIM



Gambar 4.23: Tampilan Sukses

23. Ekspor ke ZIP dan kirimkan ke e-Learning

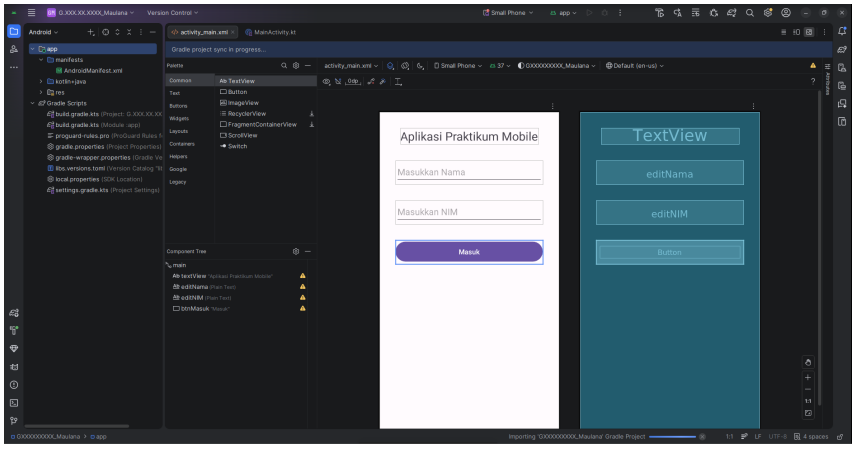
Bab 5

Fungsi dan OOP Kotlin

Praktikum ini bertujuan untuk memperkenalkan konsep dasar fungsi dan pemrograman berorientasi objek (Object-Oriented Programming/OOP) menggunakan bahasa Kotlin dalam pengembangan aplikasi Android. Mahasiswa akan mempelajari bagaimana memecah logika program ke dalam fungsi yang terstruktur serta membangun class dan object untuk merepresentasikan data secara sistematis. Melalui studi kasus sederhana berupa input data NIM dan Nama, praktikum ini menekankan pentingnya validasi input, modularisasi kode, serta pemanfaatan method dalam class untuk menghasilkan output yang lebih terorganisasi. Dengan pendekatan ini, diharapkan pemahaman terhadap struktur program yang baik dan prinsip dasar OOP dapat diterapkan pada pengembangan aplikasi yang lebih kompleks.

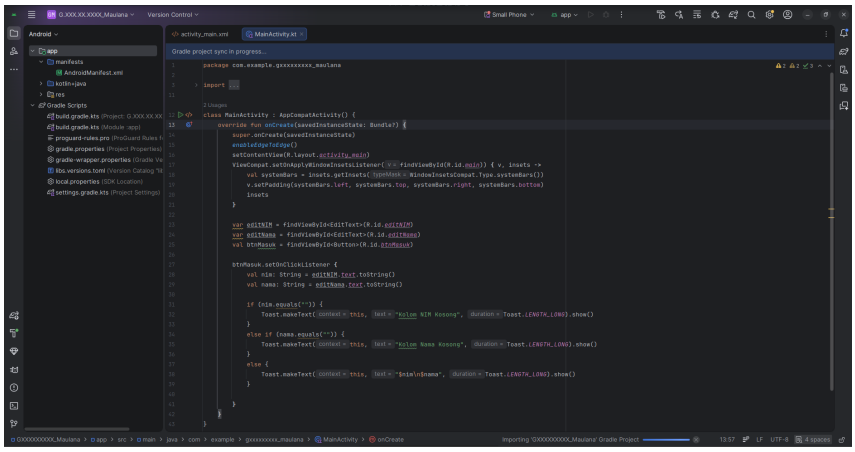
5.1 Tutorial

1. Buka kembali Android Studio dan projek yang sudah dibuat di pertemuan sebelumnya



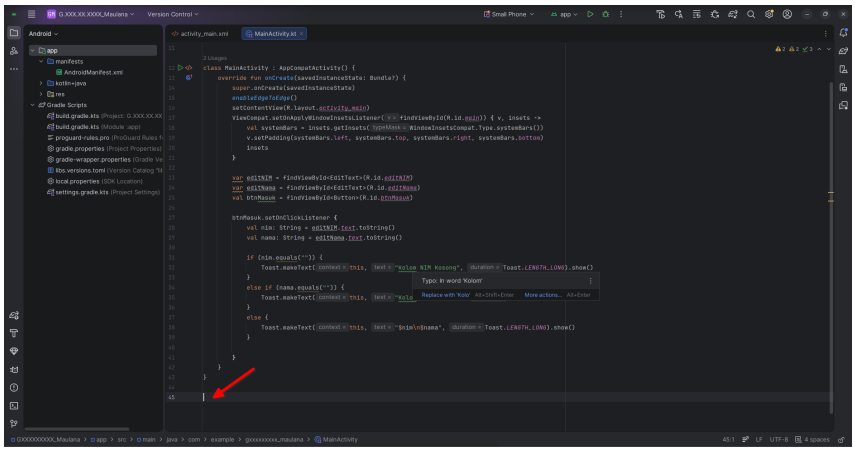
Gambar 5.1: Tampilan Android Studio dan Projek Sebelumnya

2. Kemudian buka file **MainActivity.kt**



Gambar 5.2: Membuat File MainActivity.kt

3. Letakkan kursor di paling akhir baris. Sehingga tidak akan masuk blok `class MainActivity`

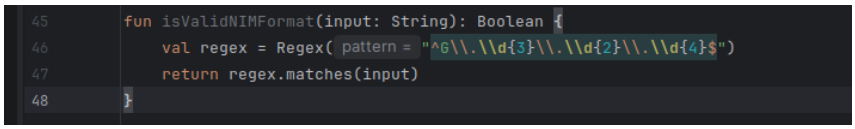


Gambar 5.3: Letak Kursor di MainActivity.kt

4. Lalu masukkan kode berikut untuk verifikasi format NIM

Potongan Kode

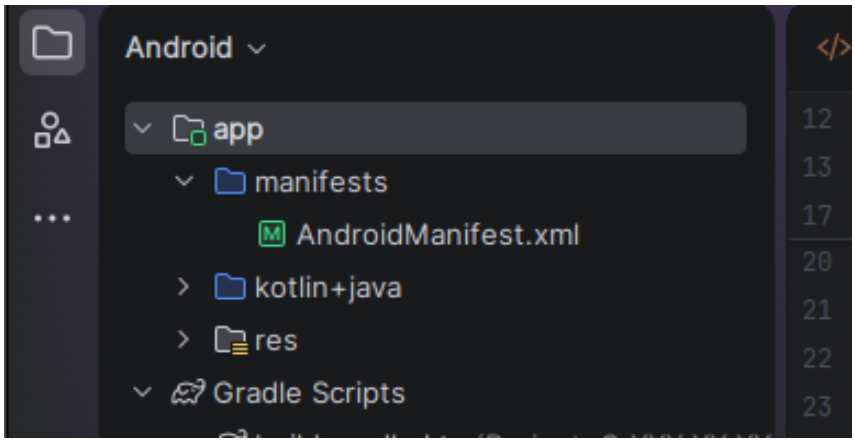
```
fun cekFormatNIM(input: String): Boolean {
  val regex = Regex("^G\\.\\d{3}\\.\\d{2}\\.\\d{4}$")
  return regex.matches(input)
}
```



Gambar 5.4: Kode Fungsi Verifikasi NIM

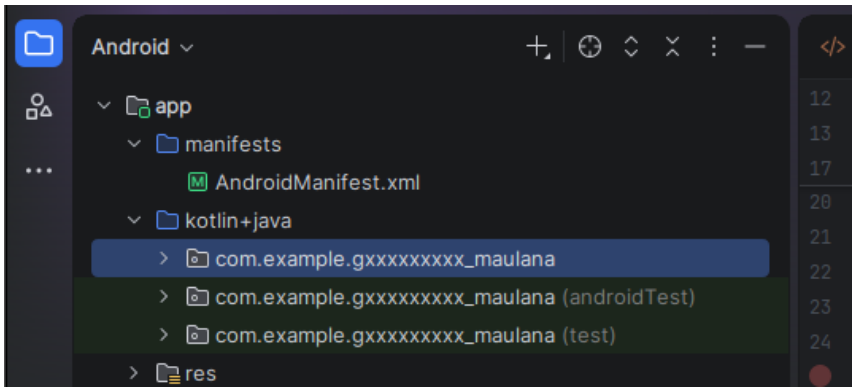
5. Lalu buat satu file Kotlin dengan cara :

   (a) klik folder **app** sehingga muncul folder isi nya. Jika tidak muncul, pastikan panel kiri dalam mode **Android**



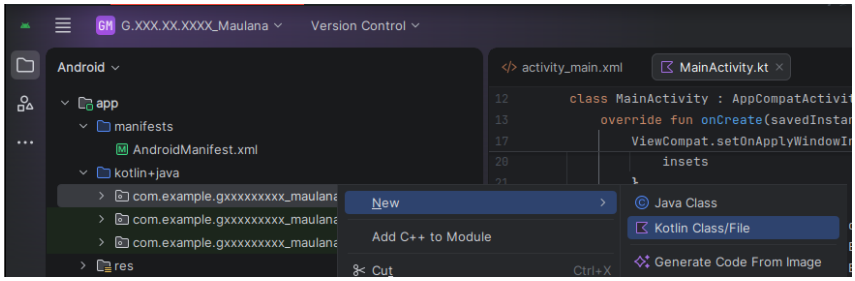
Gambar 5.5: Membuka Folder app

(b) Buka folder **kotlin+java**



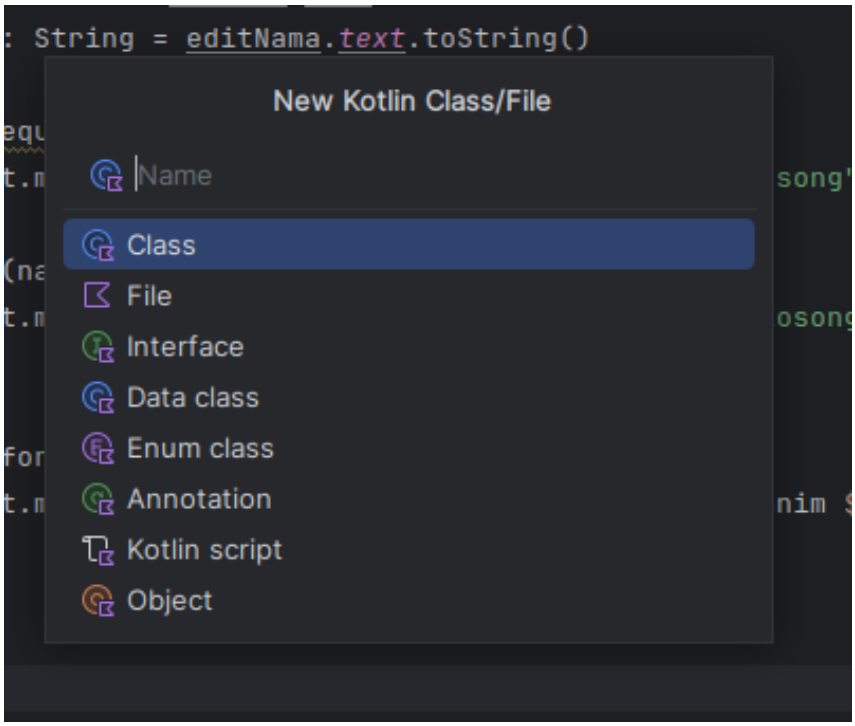
Gambar 5.6: Membuka Folder kotlin+java

(c) klik kanan folder tanpa highlight hijau dan bukan bagian dari **Test** maupun androidTest, dan pilih **New** → **Kotlin Class/File**



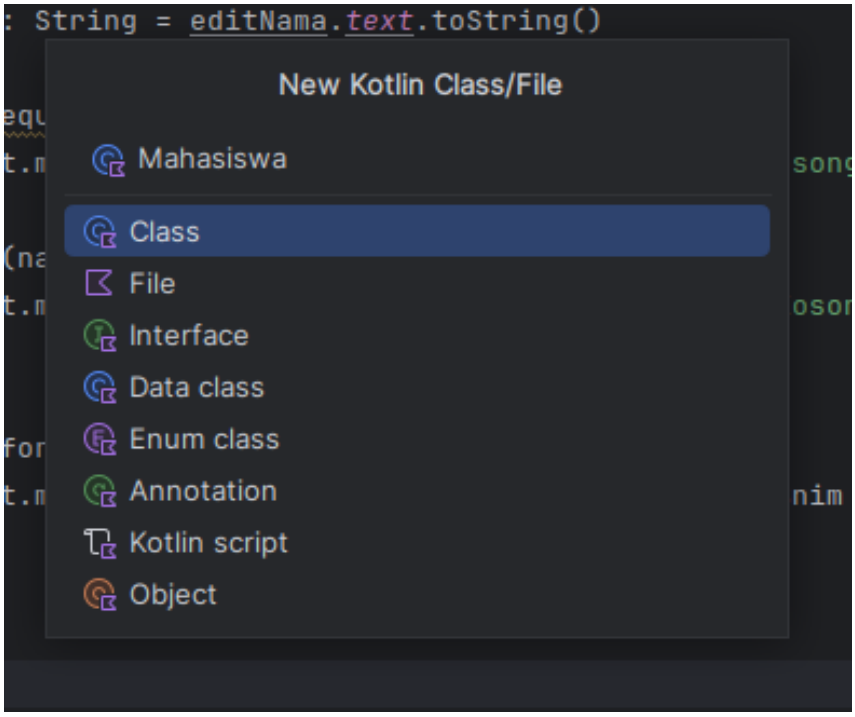
Gambar 5.7: Membuat File Kotlin

(d) Maka akan muncul window kecil



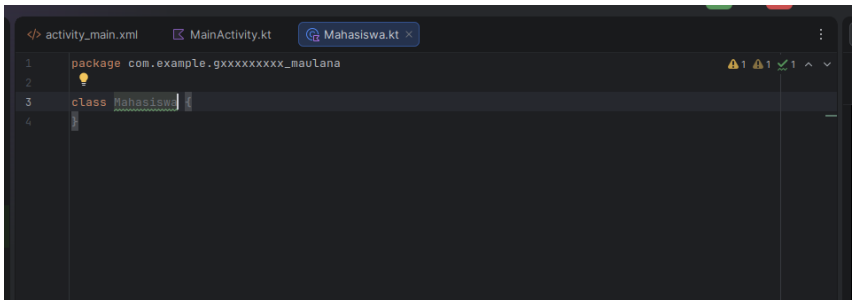
Gambar 5.8: Window Nama File

(e) Pastikan **Class** sudah terpilih, dan masukkan nama **Mahasiswa** lalu **Enter**



Gambar 5.9: Window Nama File

(f) Android Studio akan membuka file tersebut secara otomatis. Lihat gambar berikut

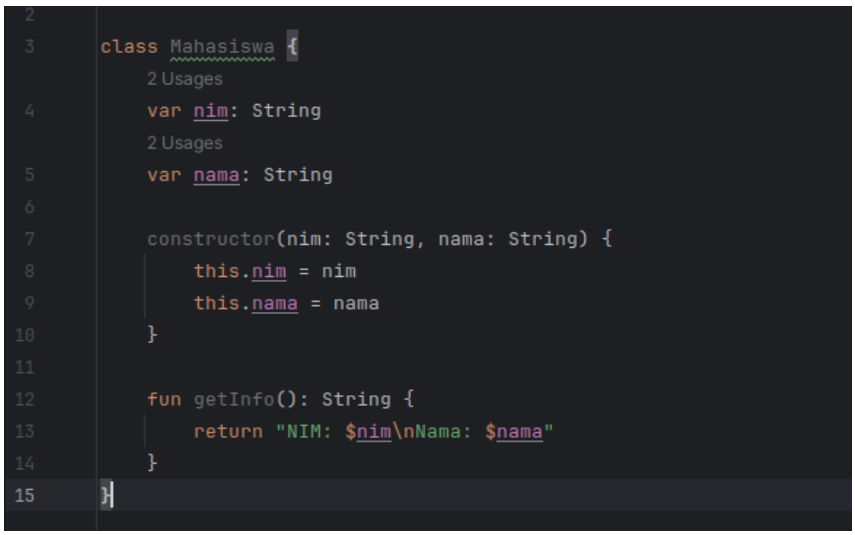


Gambar 5.10: File Mahasiswa.kt

(g) Abaikan `package .....` Di dalam blok `class Mahasiswa`. Masukkan kode berikut:

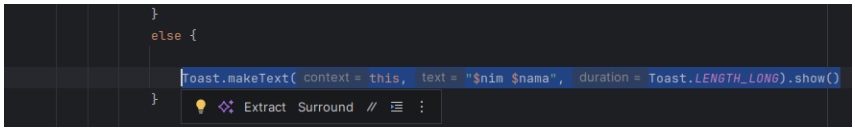
Potongan Kode

```
var nim: String
var nama: String

constructor(nim: String, nama: String) {
  this.nim = nim
  this.nama = nama
}

fun getInfo(): String {
  return "NIM: $nim\nNama: $nama"
}
```



Gambar 5.11: Kode Class Mahasiswa

6. Berikutnya adalah memberikan kode untuk memastikan format NIM sudah sesuai. Di blok **else** paling akhir. Hapus kode blok `Toast.makeText()`

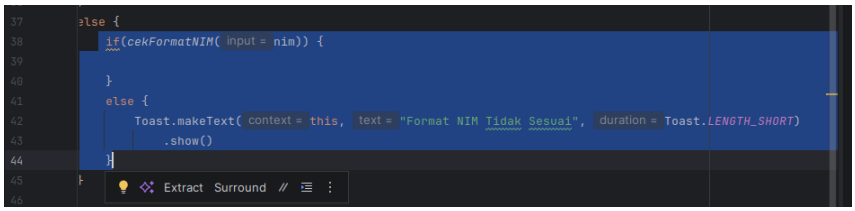


Gambar 5.12: Hapus Isi Blok Else

7. Lalu masukkan kode **if else** untuk verifikasi format NIM seperti berikut di dalam blok `else`

Potongan Kode

```
if(cekFormatNIM(nim)) {

}
else {
  Toast.makeText(this, "Format NIM Tidak Sesuai", Toast.LENGTH_SHORT)
.show()
}
```

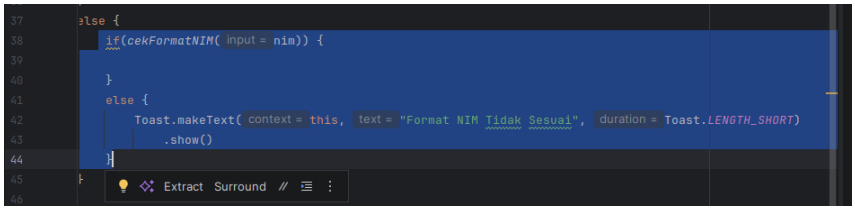


Gambar 5.13: Isi Blok Else

8. Lanjutkan kode di dalam blok **if** yang kosong tersebut dengan kode berikut

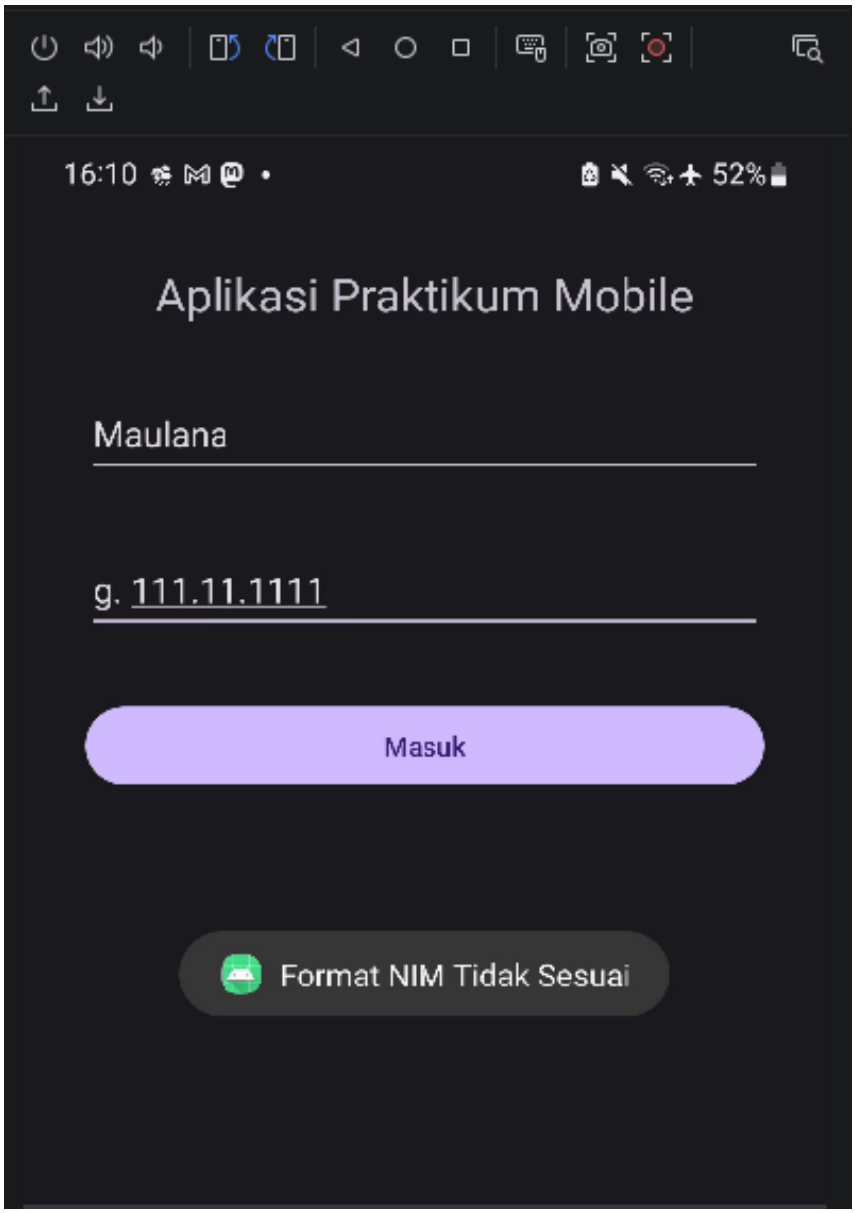
Potongan Kode

```
var mhs = Mahasiswa(nim=nim, nama=nama)
Toast.makeText(this, "Class:\nNIM:$mhs.nim\nNama:$mhs.nama", Toast.LENGTH_SHORT)
.show()
```

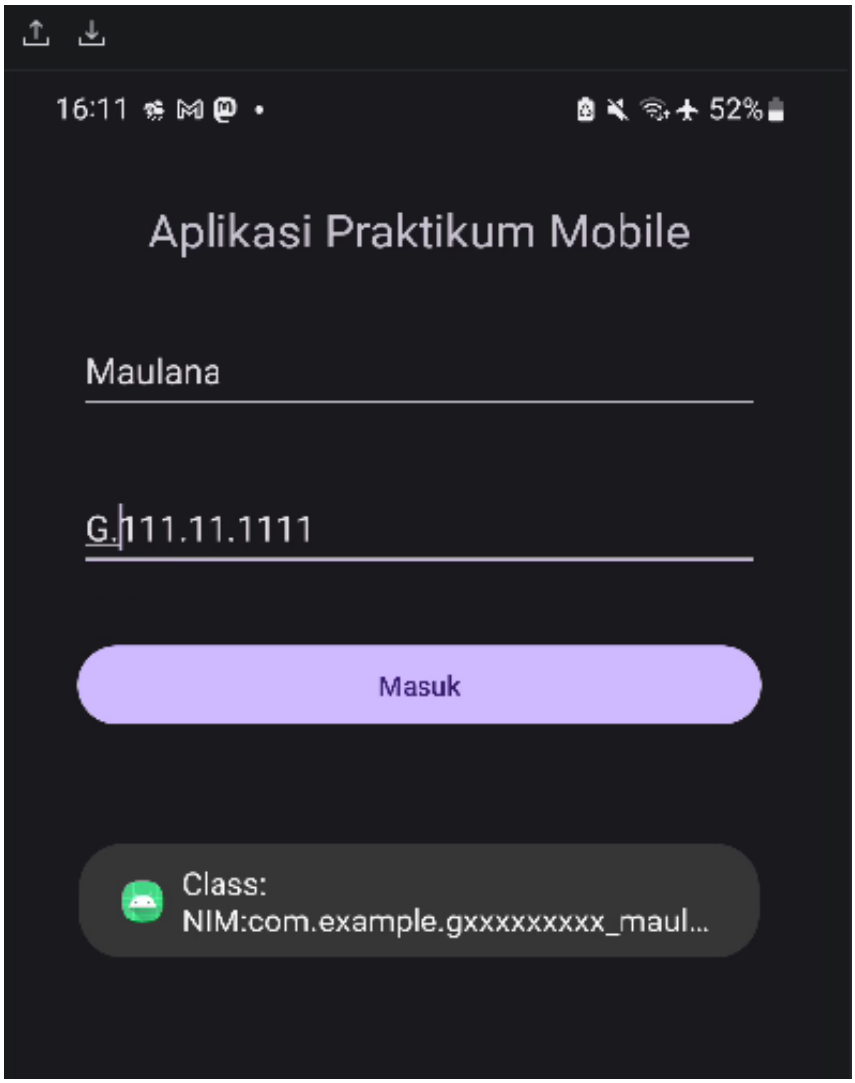


Gambar 5.14: Isi Blok IF

9. Jalankan aplikasi untuk menguji apakah pengecekan format NIM dan enkapsulasi Class terhadap data mahasiswa berhasil



Gambar 5.15: Error Format NIM Salah



Gambar 5.16: Class Berhasil

10. Jika yang muncul bukan nama melainkan nama **package**. Maka enkapsulasi oleh `Class Mahasiswa` telah berhasil. Untuk bisa mengambil data dari kelas harus menggunakan metode dari kelas itu sendiri yaitu `getInfo()`

Bab 6

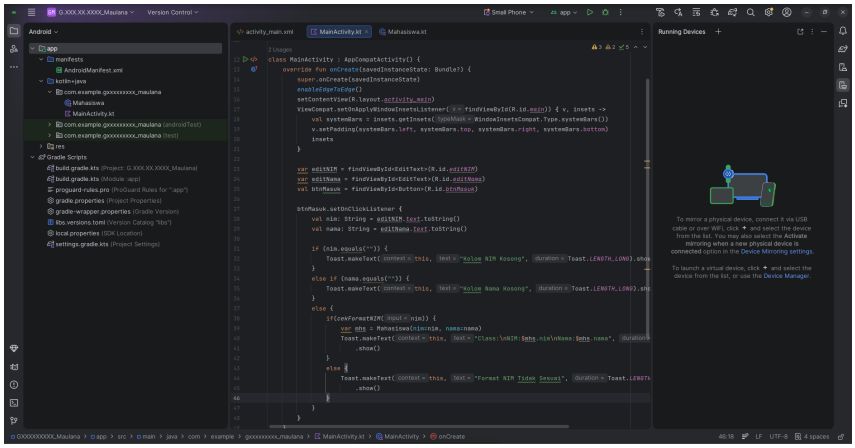
Event Handling dan Navigasi

6.1 Pendahuluan

Pemrograman aplikasi mobile berbasis Android merupakan salah satu bidang yang berkembang sangat pesat dalam teknologi informasi modern. Dalam pengembangan aplikasi Android, kemampuan menangani interaksi pengguna dan perpindahan antar halaman menjadi konsep fundamental yang wajib dipahami oleh pengembang. Event Handling digunakan untuk menangani berbagai aksi pengguna seperti menekan tombol, memasukkan data, maupun memilih komponen antarmuka, sedangkan Navigasi digunakan untuk mengatur perpindahan antar Activity atau halaman aplikasi agar alur penggunaan menjadi terstruktur dan interaktif. Pada materi ini, implementasi Event Handling dan Navigasi dilakukan menggunakan bahasa Kotlin pada lingkungan pengembangan Android Studio. Melalui praktikum ini, mahasiswa diharapkan mampu memahami dasar pengelolaan event, validasi input pengguna, penggunaan komponen Intent, serta mekanisme pengiriman data antar Activity sebagai fondasi dalam membangun aplikasi Android yang dinamis dan responsif.

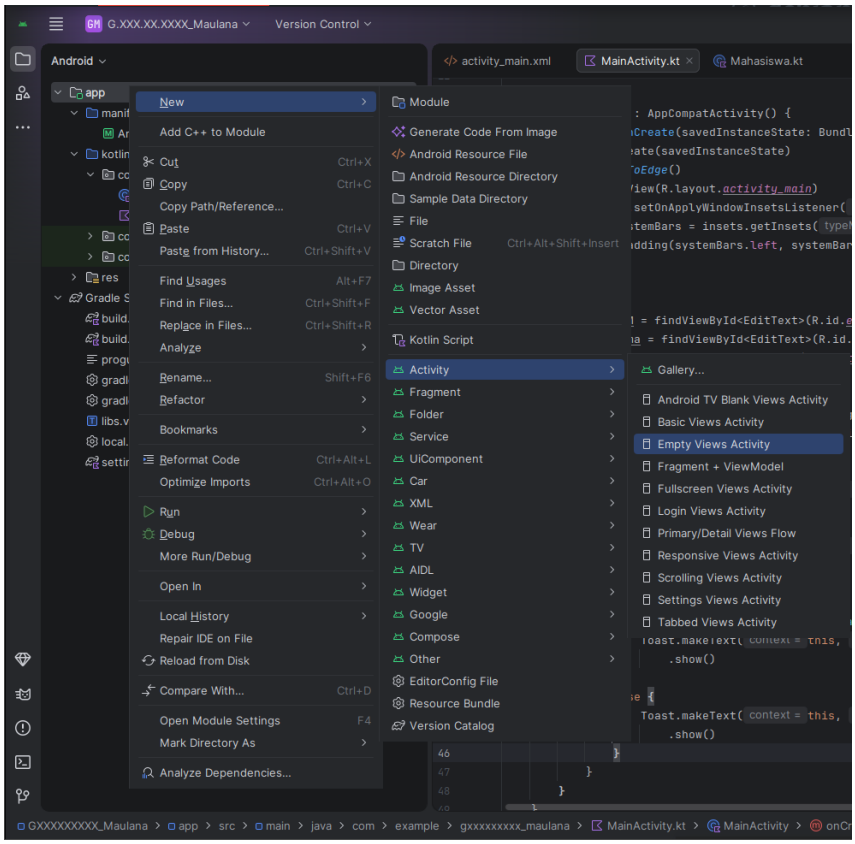
6.2 Tutorial

1. Buka kembali projek yang sudah dikerjakan sebelumnya.



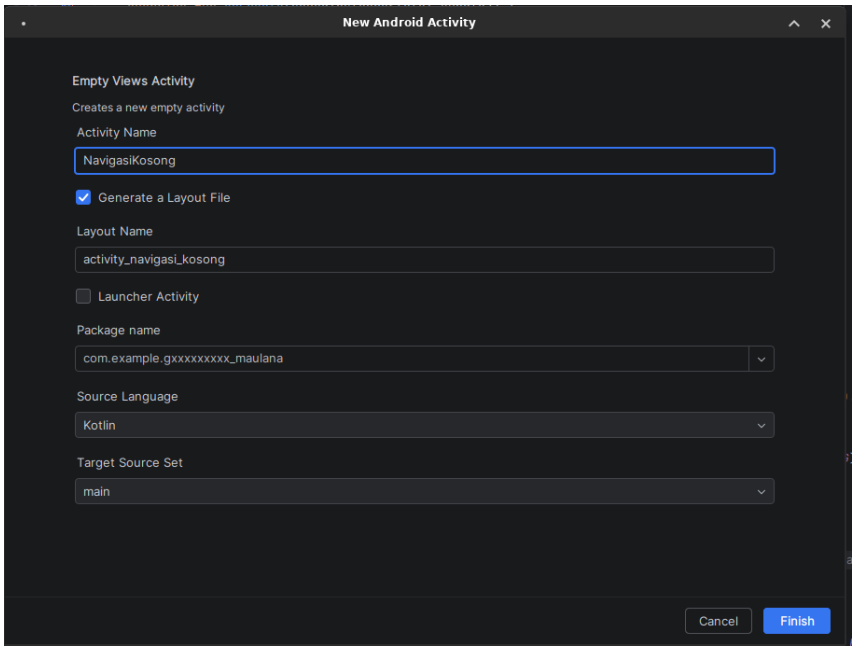
Gambar 6.1: Buka Projek Sebelumnya

2. Untuk bisa melakukan navigasi ke halaman berikutnya. Di bagian panel kiri, klik kanan **app** → Pilih **New** → Activity → **Empty View Activity**



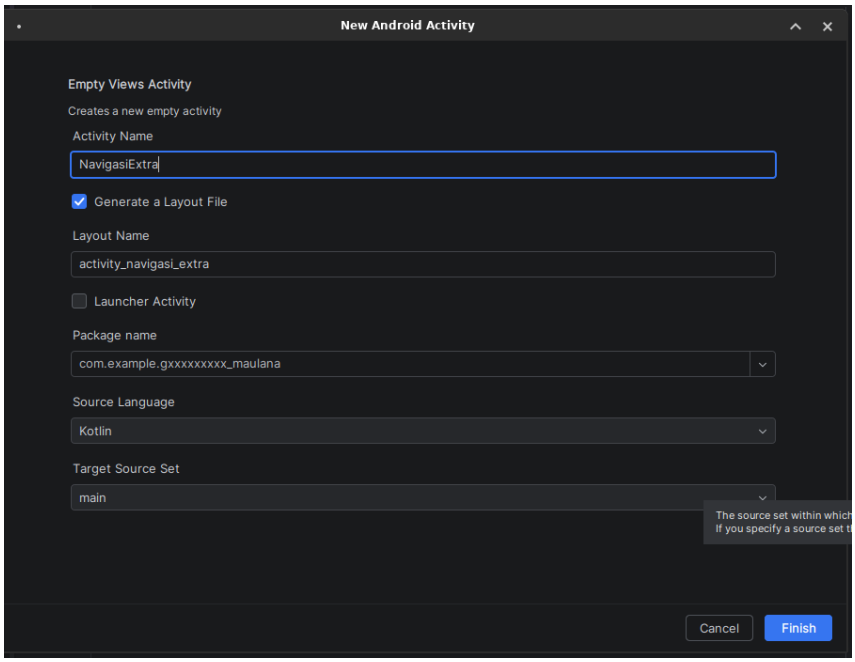
Gambar 6.2: Tahapan membuat Activity Baru

3. Android Studio akan menampilkan window pembuatan Activity baru. Masukkan nama aktivitas dengan **NavigasiKosong**



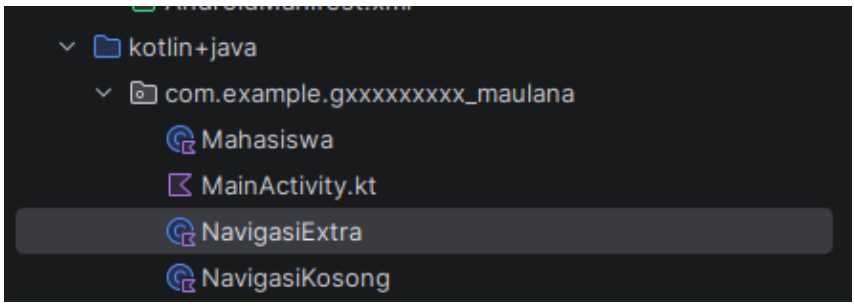
Gambar 6.3: Activity NavigasiKosong

4. Buat satu activity lagi dengan nama **NavigasExtra**



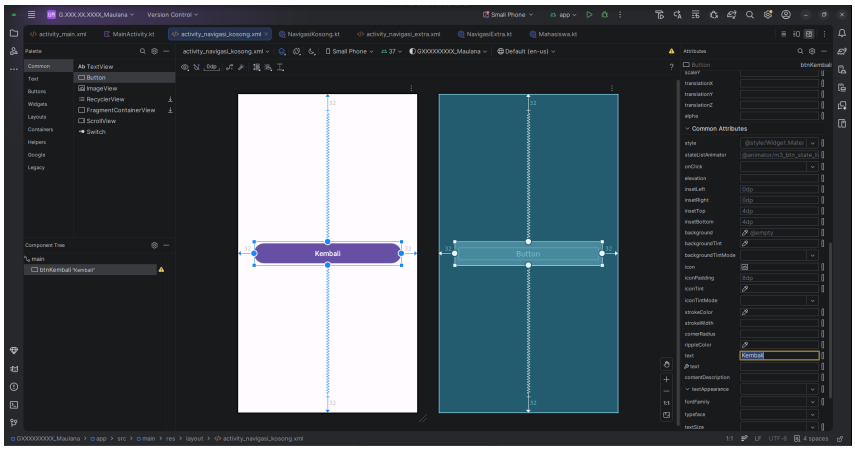
Gambar 6.4: Activity NavigasiExtra

5. Jika berhasil, di panel kiri akan ada 2 activity baru dengan nama **NavigasiKosong** dan **NavigasiExtra**



Gambar 6.5: Activity Telah Dibuat

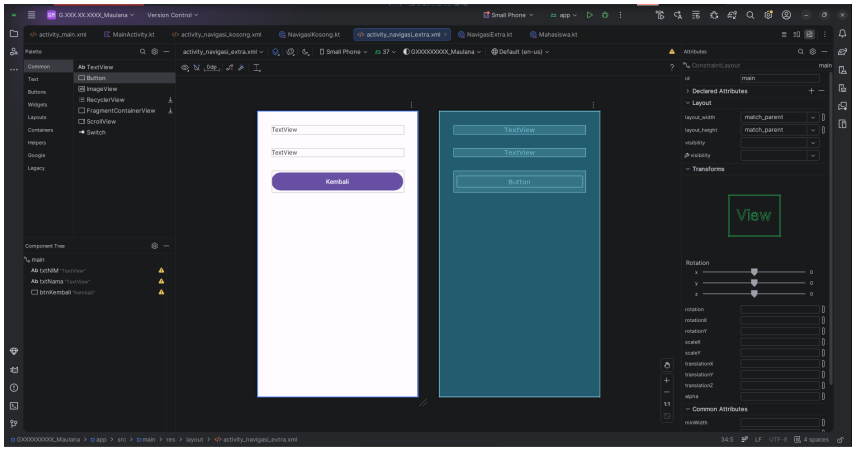
6. Buka file layout **NavigasiKosong** di folder **res** → **layout** → **activity_navigasi_kosong.xml**. Lalu tambahkan satu tombol **Kembali** dengan ID **btnKembali**. Atur **attribute Constraint** agar bisa di tengah-tengah



Gambar 6.6: Activity dan Button

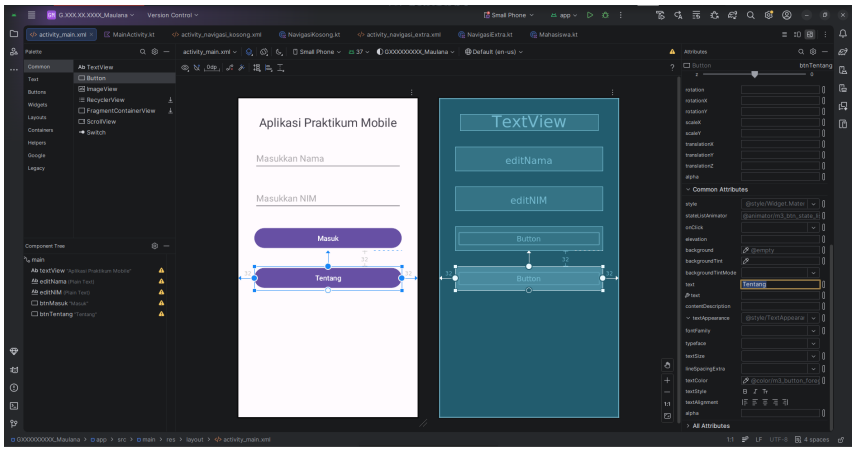
7. Buka file layout **NavigasiExtra** dengan nama **activity_navigasi_extra.xml** dan masukkan komponen dan konfigurasikan ID sebagai berikut

   - TextView : txtNIM
   - TextView : txtNama
   - Button : btnKembali



Gambar 6.7: Activity Extra, TextView dan Button

8. Jika sudah selesai dengan objek UI nya. Buka file layout **activity_main.xml**, lalu masukkan **Button** dengan tulisan **Tentang** dan **ID btnTentang**



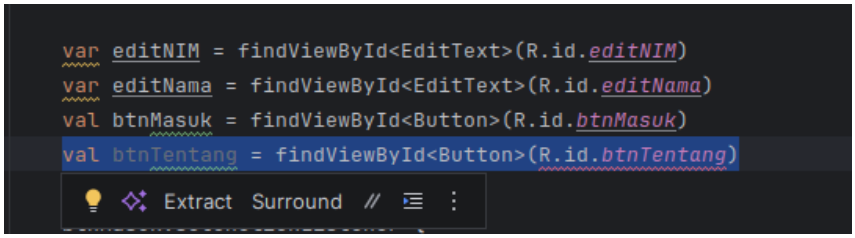
Gambar 6.8: Tambah Button Baru

9. Berikutnya adalah memrogram Button tersebut. Buka file **MainActivity.kt** tambahkan kode-kode berikut

   - Inisialisasi Button Tentang

Potongan Kode

```
val btnTentang = findViewById<Button>(R.id.btnTentang)
```

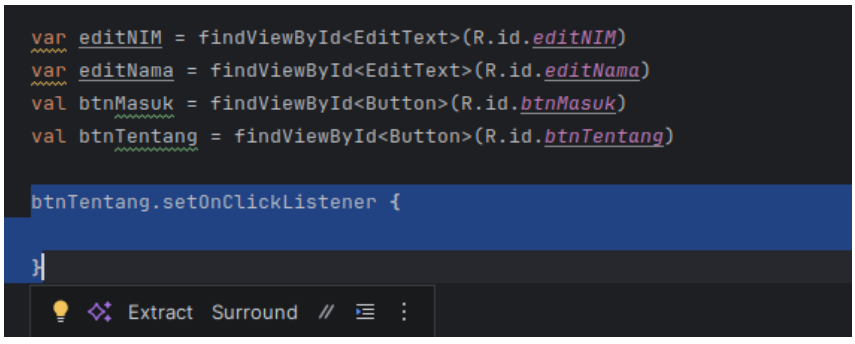


Gambar 6.9: Inisialisasi btnTentang

   - Tambahkan Listener pada kode **btnTentang**

Potongan Kode

```
btnTentang.setOnClickListener {

}
```

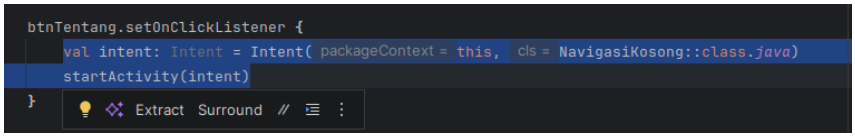


Gambar 6.10: Tambah Listener btnTentang

   - Tambahkan kode **Intent** untuk navigasi ke **NavigasiKosong**

Potongan Kode

```
val intent: Intent = Intent(this, NavigasiKosong::class.java)
startActivity(intent)
```

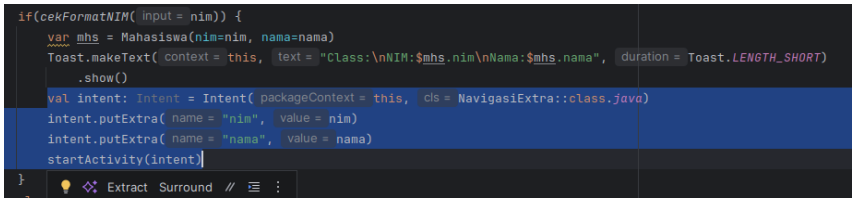


Gambar 6.11: Tambah Listener btnTentang

- Tambahkan **Intent** di **btnMasuk** di blok `if(cekFormatNIM(nim))` setelah Toast

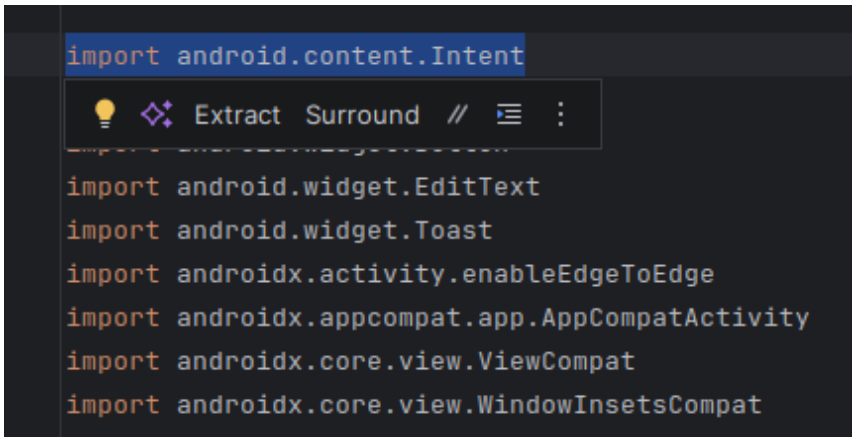
Potongan Kode

```
val intent: Intent = Intent(this, NavigasiExtra::class.java)
intent.putExtra("nim", nim)
intent.putExtra("nama", nama)
startActivity(intent)
```



Gambar 6.12: Tambah Listener btnTentang

- Tambahkan import untuk **Intent**

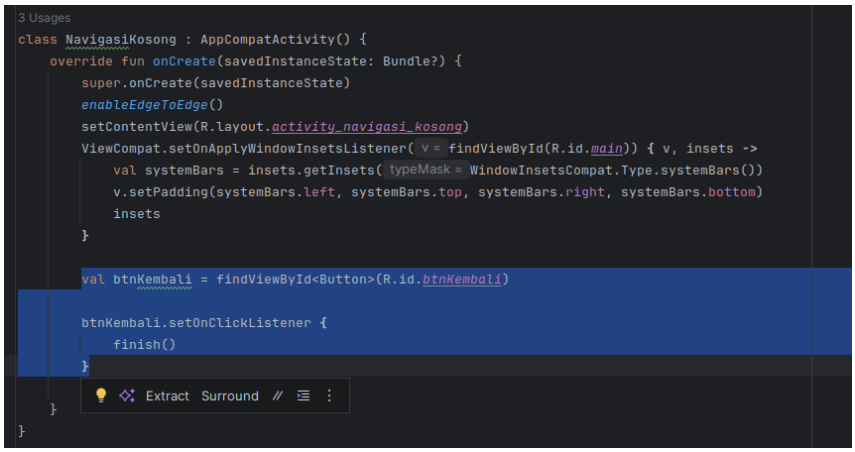


Gambar 6.13: Tambah Import Intent

10. Jika sudah, buka file **NavigasiKosong.kt**, masukkan kode berikut untuk inisialisasi **btnKembali** dan kode kembali

Potongan Kode

```
val btnKembali = findViewById<Button>(R.id.btnKembali)

btnKembali.setOnClickListener {
        finish()
}
```

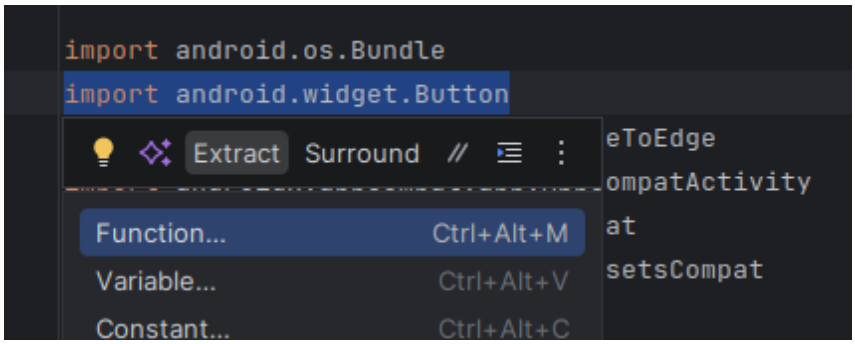


Gambar 6.14: Tambah Listener btnKembali

11. Tambahkan import **Button** di bagian atas

Potongan Kode

```
import android.widget.Button
```



Gambar 6.15: Tambah Import Button

12. Buka **NavigasiExtra.kt** dan masukkan kode berikut

Potongan Kode

```
var txtNIM = findViewById<TextView>(R.id.txtNIM)
var txtNama = findViewById<TextView>(R.id.txtNama)
val btnKembali = findViewById<Button>(R.id.btnKembali)

// Ambil Data dari Intent
val dataNIM = intent.getStringExtra("nim")
val dataNama = intent.getStringExtra("nama")

// Set Data ke TextView
txtNIM.text = dataNIM
txtNama.text = dataNama

btnKembali.setOnClickListener {
        finish()
}
```

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    enableEdgeToEdge()
    setContentView(R.layout.activity_navigasi_extra)
    ViewCompat.setOnApplyWindowInsetsListener( v = findViewById(R.id.main)) { v, insets ->
        val systemBars = insets.getInsets( typeMask = WindowInsetsCompat.Type.systemBars())
        v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
        insets
    }

    var txtNIM = findViewById<TextView>(R.id.txtNIM)
    var txtNama = findViewById<TextView>(R.id.txtNama)
    val btnKembali = findViewById<Button>(R.id.btnKembali)

    val dataNIM = intent.getStringExtra( name = "nim")
    val dataNama = intent.getStringExtra( name = "nama")

    txtNIM.text = dataNIM
    txtNama.text = dataNama

    btnKembali.setOnClickListener {
        finish()
    }
```

Gambar 6.16: Tambah Kode TextView, Intent, dan Button

13. Tambahkan import yang diperlukan

Potongan Kode

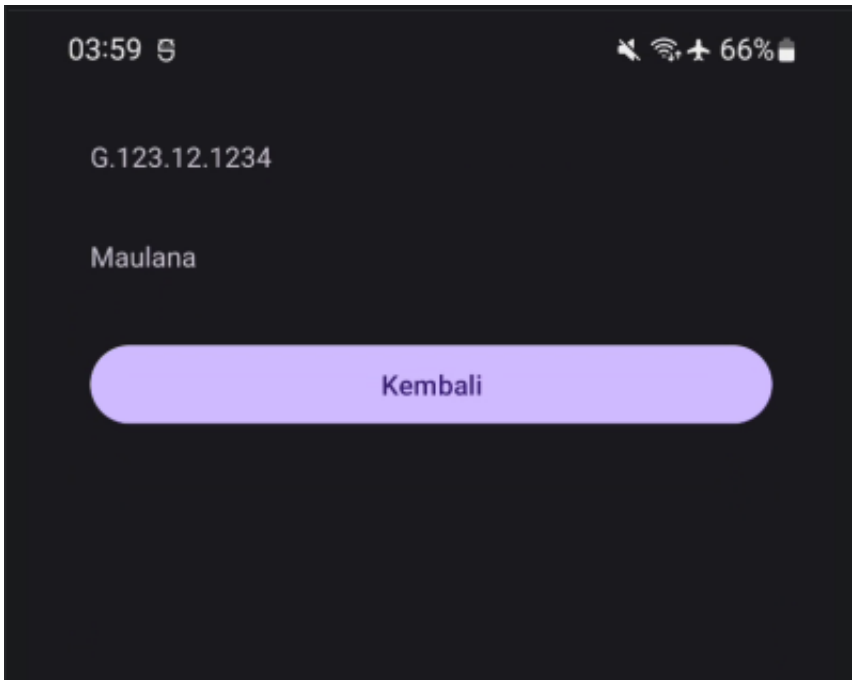
```
import android.widget.Button
import android.widget.TextView
```

```
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
                                    eToEdge
                                    ompatActivity
import androidx.core.view.ViewCompat
import androidx.core.view.WindowInsetsCompat
```

Extract Surround

Gambar 6.17: Tambah Kode Import

14. Jalankan aplikasi dan coba tombol-tombol yang sudah dibuat
    - btnTentang → NavigasiKosong
    - btnMasuk → NavigasiExtra + Data



Gambar 6.18: Hasil

15. Kirimkan **Export to ZIP** ke e-Learning

Bab 7

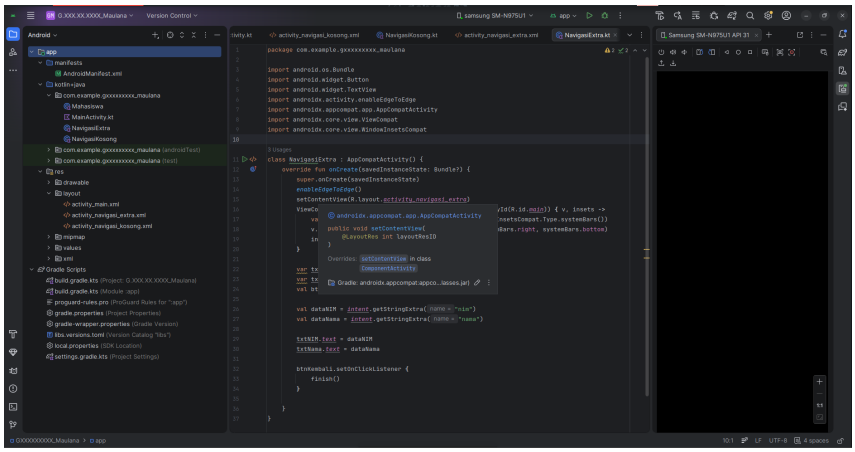
Penyimpanan Data Lokal : SQLite

7.1 Pendahuluan

Praktikum ini bertujuan untuk memberikan pemahaman dasar mengenai implementasi basis data lokal menggunakan SQLite pada aplikasi Android dengan memanfaatkan Android Studio. Mahasiswa akan mempelajari proses pembuatan database, tabel, serta penerapan operasi CRUD (Create, Read, Update, Delete) menggunakan bahasa pemrograman Kotlin. Selain itu, praktikum ini juga melatih kemampuan dalam menghubungkan antarmuka aplikasi dengan sistem penyimpanan data lokal sehingga aplikasi mampu menyimpan dan menampilkan data secara dinamis. Melalui kegiatan praktikum ini, diharapkan mahasiswa mampu memahami konsep pengelolaan data pada perangkat bergerak serta mampu mengembangkan aplikasi Android sederhana yang terintegrasi dengan SQLite sebagai fondasi pengembangan aplikasi mobile berbasis data.

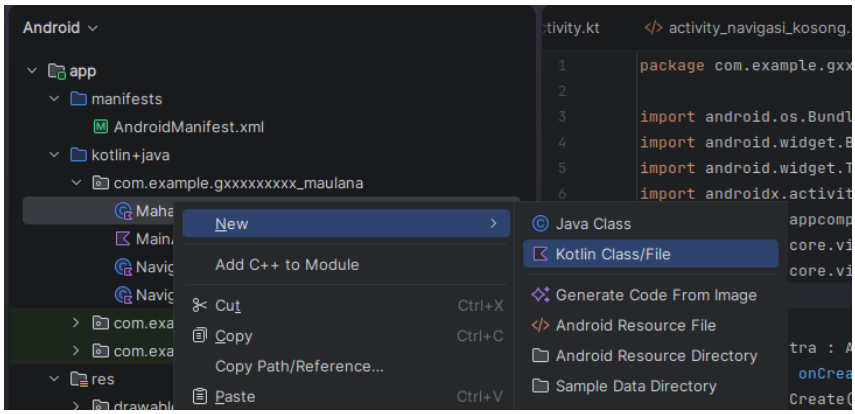
7.2 Tutorial

1. Buka projek Android Studio



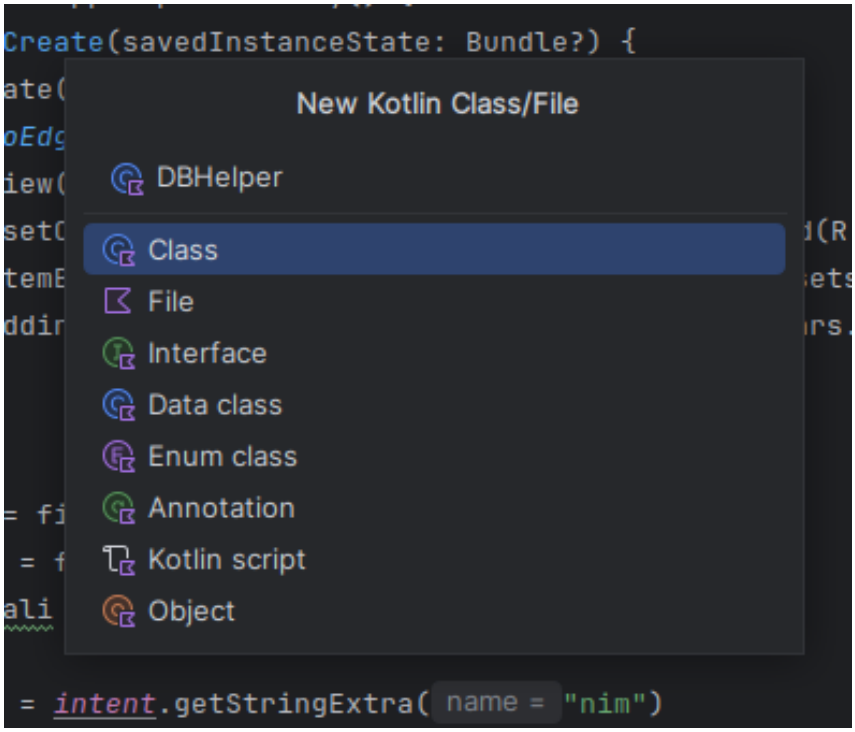
Gambar 7.1: Buka Projek

2. Di bagian panel kiri, cari file **Mahasiswa.kt**. Klik Kanan file tersebut → **New** → **Kotlin Class/File**



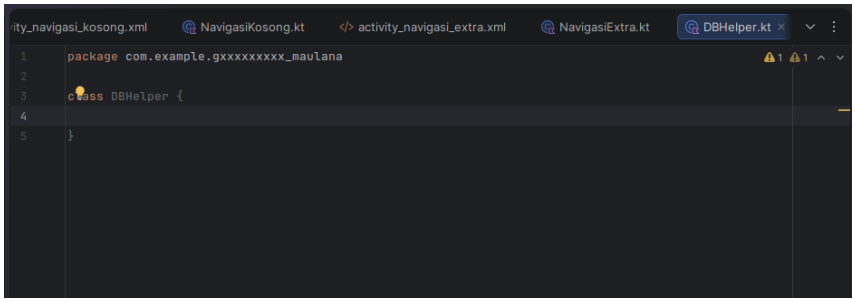
Gambar 7.2: Buat File Kotlin

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



Gambar 7.3: Nama File Kotlin

4. File **DBHelper** telah dibuat

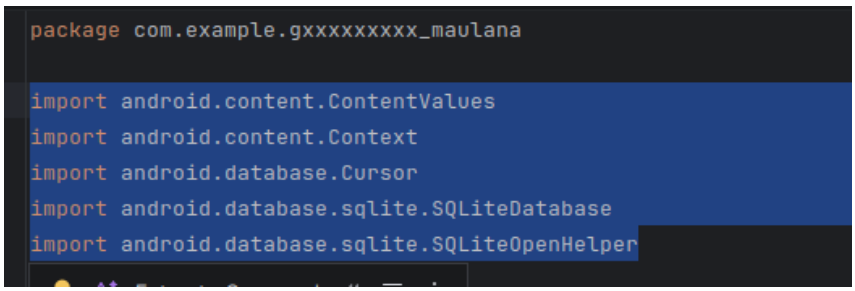


Gambar 7.4: File DBHelper

5. Masukkan kode **DBHelper** berikut

Potongan Kode

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

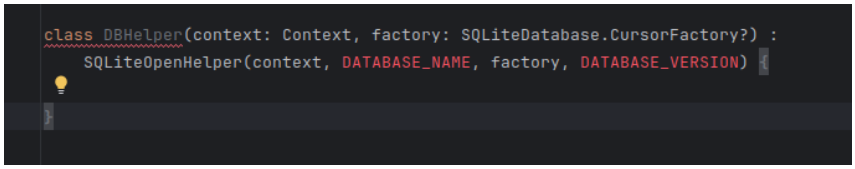


Gambar 7.5: Tambah Kode Import DBHelper

6. Ubah `class DBHelper`, menjadi berikut

Potongan Kode

```
class DBHelper(context: Context, factory: SQLiteDatabase.CursorFactory?) : {

}
```



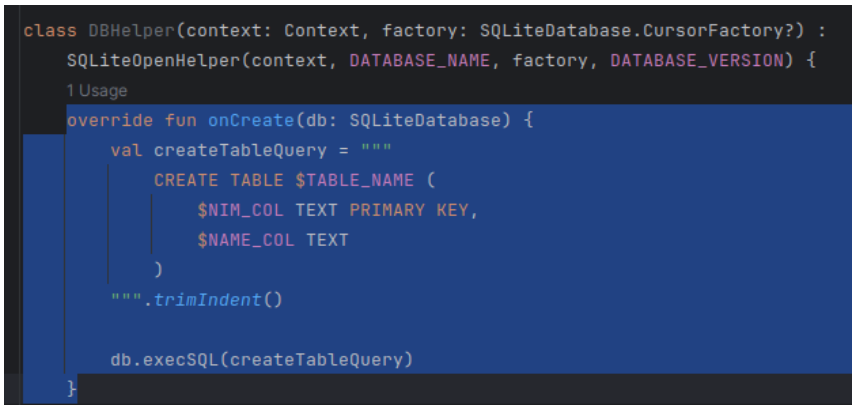
Gambar 7.6: Ubah kode Class

7. Di dalam nya masukkan kode berikut

- **Create Database**

Potongan Kode

```
override fun onCreate(db: SQLiteDatabase) {
        val createTableQuery = """
        CREATE TABLE $TABLE_NAME (
        $NIM_COL TEXT PRIMARY KEY,
        $NAME_COL TEXT
        )
        """.trimIndent()

        db.execSQL(createTableQuery)
}
```

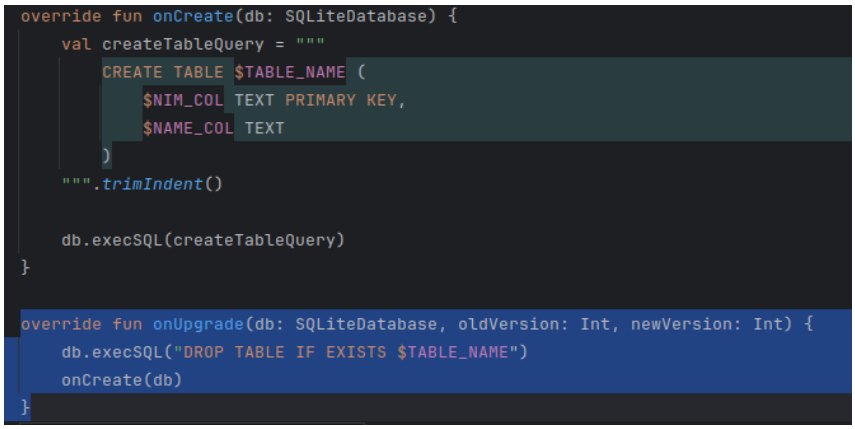


Gambar 7.7: Kode Create Database

- **Upgrade Database**

Potongan Kode

```
override fun onUpgrade(db: SQLiteDatabase, oldVersion: Int,
        newVersion: Int) {
        db.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
}
```

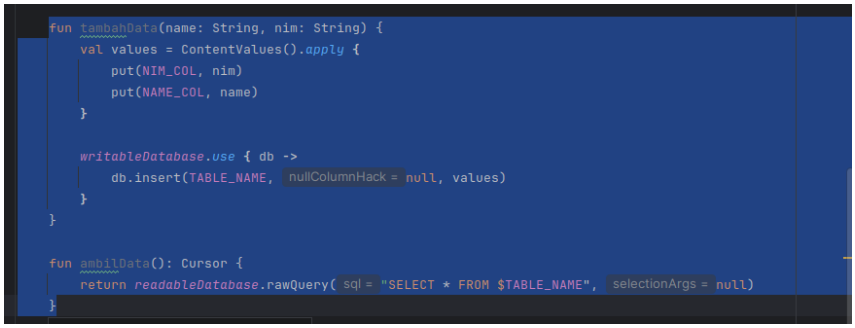


Gambar 7.8: Kode Upgrade Database

- Fungsi **tambahData** dan **ambilData**

Potongan Kode

```
fun tambahData(name: String, nim: String) {
        val values = ContentValues().apply {
                put(NIM_COL, nim)
                put(NAME_COL, name)
        }

        writableDatabase.use { db ->
                db.insert(TABLE_NAME, null, values)
        }
}

fun ambilData(): Cursor {
        return readableDatabase.rawQuery("SELECT * FROM $TABLE_NAME", null)
}
```



Gambar 7.9: Kode Fungsi

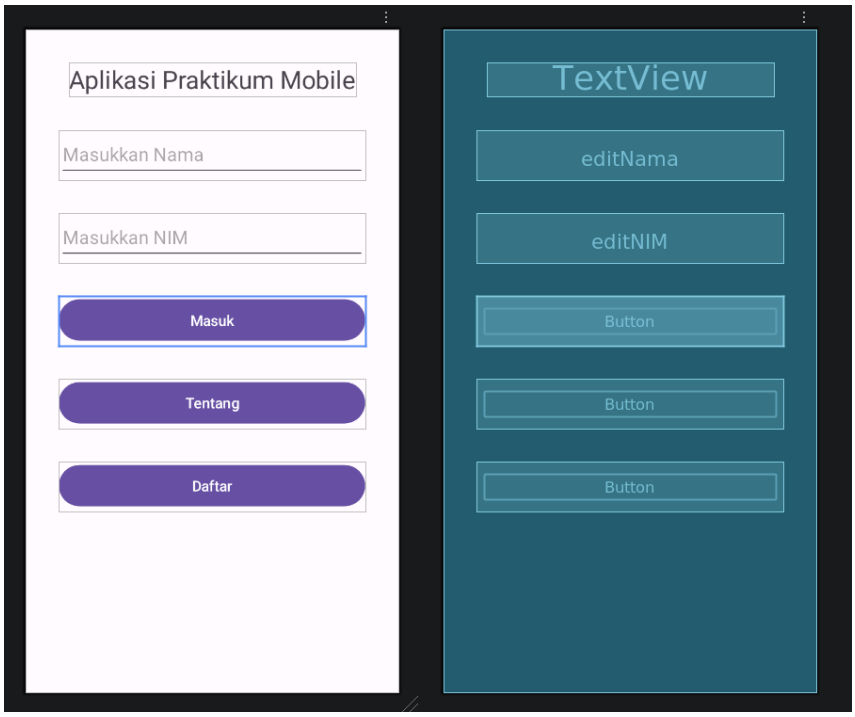
- Kode **Companion**

Potongan Kode

```
companion object {
        private const val DATABASE_NAME = "mahasiswa"
        private const val DATABASE_VERSION = 1
        const val TABLE_NAME = "mahasiswa"
        const val NIM_COL = "nim"
        const val NAME_COL = "name"
}
```

Gambar 7.10: Kode Companion

8. Buka layout **activity_main.xml**, dan tambahkan satu tombol dengan nama **Daftar** dan **btnDaftar**



Gambar 7.11: Tambah Button Daftar

9. Inisialisasikan dan Tambahkan Listener

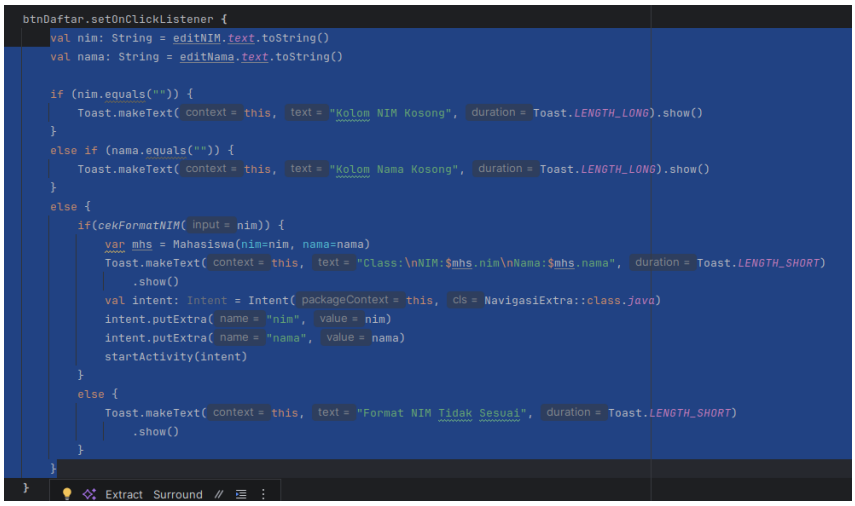
Potongan Kode

```
val btnDaftar = findViewById<Button>(R.id.btnDaftar)

btnDaftar.setOnClickListener {

}
```

```
var editNIM = findViewById<EditText>(R.id.editNIM)
var editNama = findViewById<EditText>(R.id.editNama)
val btnMasuk = findViewById<Button>(R.id.btnMasuk)
val btnTentang = findViewById<Button>(R.id.btnTentang)
val btnDaftar = findViewById<Button>(R.id.btnDaftar)

btnDaftar.setOnClickListener {

}
```

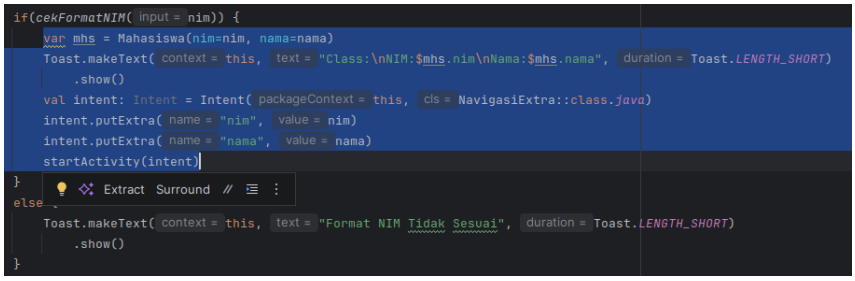
Gambar 7.12: Inisialisasi Kode Button Daftar

10. Kopi kode verifikasi dari **btnMasuk** dan tempelkan di **btnDaftar**

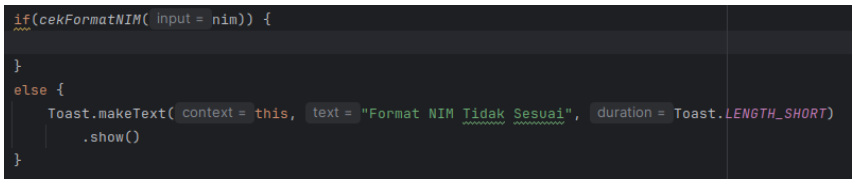


Gambar 7.13: Kopi Kode btnMasuk ke btnDaftar

11. Hapus kode yang ada di dalam `if(cekFormatNIM(nim))` di **btnDaftar**



Gambar 7.14: Hapus Kode



Gambar 7.15: Hapus Kode

12. Masukkan kode registrasi database sebagai berikut

Potongan Kode

```
db.tambahData(nama, nim)
Toast.makeText(this, "Sukses", Toast.LENGTH_LONG).show()

editNIM.text.clear()
editNama.text.clear()
```

```
if(cekFormatNIM( input = nim)) {
    db.tambahData( name = nama, nim)
    Toast.makeText( context = this, text = "Sukses", duration = Toast.LENGTH_LONG).show()

    editNIM.text.clear()
    editNama.text.clear()
}
else {
    Toast.makeText( context = this, text = "Format NIM Tidak Sesuai", duration = Toast.LENGTH_SHORT)
        .show()
}
```

Extract Surround

Gambar 7.16: Kode Registrasi Database

13. Tambahkan inisialisasi database setelah **Button**

Potongan Kode

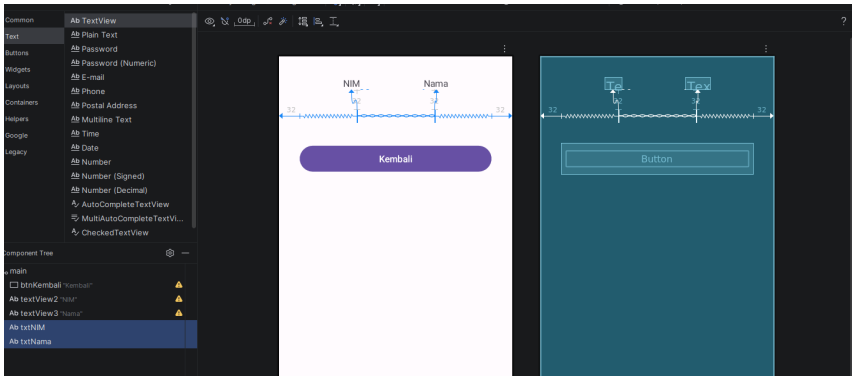
```
val db = DBHelper(this, null)
```

```
var editNIM = findViewById<EditText>(R.id.editNIM)
var editNama = findViewById<EditText>(R.id.editNama)
val btnMasuk = findViewById<Button>(R.id.btnMasuk)
val btnTentang = findViewById<Button>(R.id.btnTentang)
val btnDaftar = findViewById<Button>(R.id.btnDaftar)

val db = DBHelper( context = this, factory = null)
```

Extract Surround

Gambar 7.17: Inisialisasi Database

14. Untuk menampilkan data, buka **activity_navigasi_kosong.xml**. Lalu tambahkan empat TextView dengan tatanan sebagai berikut
    - NIM : abaikan ID
    - Nama : abaikan ID
    - (Kosong) : txtNIM
    - (Kosong) : txtNama



Gambar 7.18: Ubah Tampilan NavigasiKosong

15. Buka file **NavigasiKosong.kt** lalu tambahkan kode berikut untuk mengambil data secara otomatis dari database

Potongan Kode

```
val txtNIM = findViewById<TextView>(R.id.txtNIM)
val txtNama = findViewById<TextView>(R.id.txtNama)
val db = DBHelper(this, null)

txtNIM.text = ""
txtNama.text = ""

val cursor = db.ambilData()

cursor.use {
 if (cursor.moveToFirst()) {
  do {
   val cursorNama = cursor.getColumnIndexOrThrow(DBHelper.NAME_COL)
   val mhsNama = cursor.getString(cursorNama)
   val cursorNIM = cursor.getColumnIndexOrThrow(DBHelper.NIM_COL)
   val mhsNIM = cursor.getString(cursorNIM)

   txtNama.append("$mhsNama\n")
   txtNIM.append("$mhsNIM\n")
   } while (cursor.moveToNext())
  }
}
```

Gambar 7.19: Kode Pengambilan dan Tampilkan Data

16. Tambahkan kode import untuk **TextView**

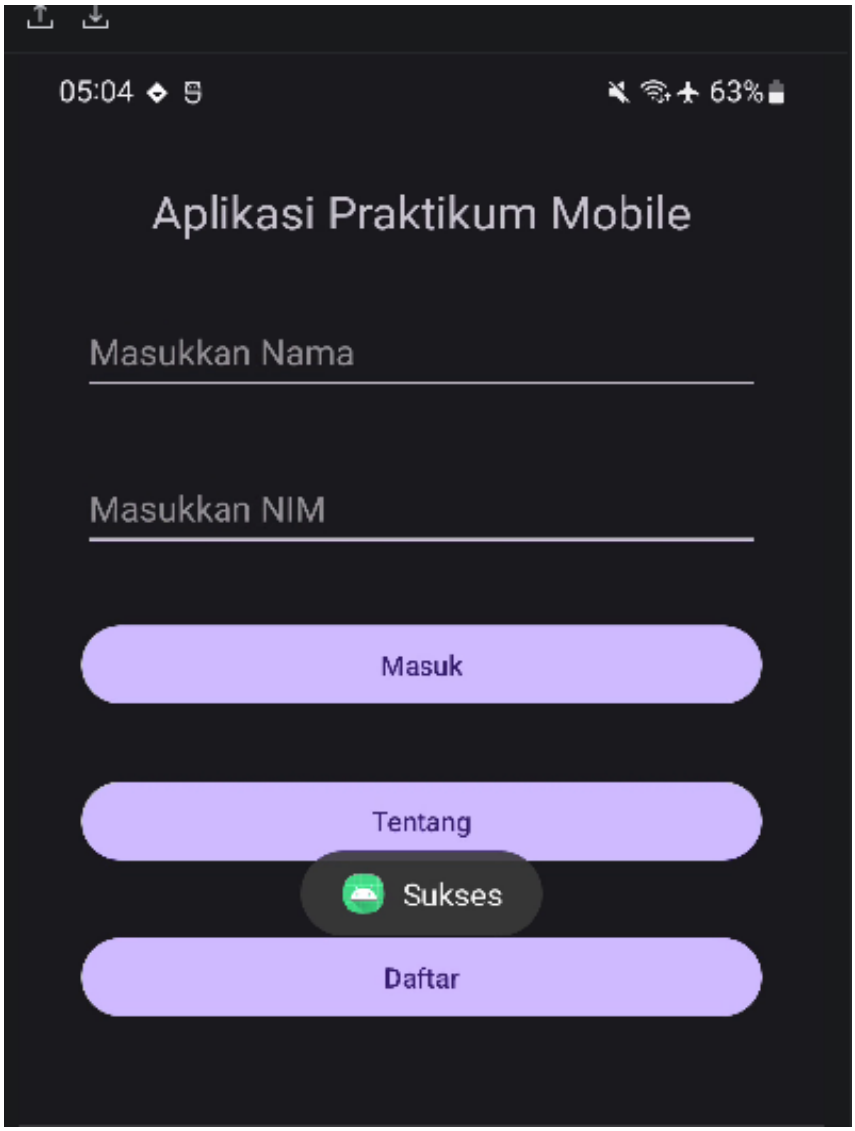
Potongan Kode

```
import android.widget.TextView
```

```
val btnKembali = findViewById<Button>(R.id.btnKembali)
val txtNIM = findViewById<TextView>(R.id.txtNIM)
val txtNama = findViewById<TextView>(R.id.txtNama)
val db = DBHelper( context = this,  factory = null)

txtNIM.text = ""
txtNama.text = ""

val cursor = db.ambilData()

cursor.use {
    if (cursor.moveToFirst()) {
        do {
            val cursorNama = cursor.getColumnIndexOrThrow( p0 = DBHelper.NAME_COL)
            val mhsNama = cursor.getString( p0 = cursorNama)
            val cursorNIM = cursor.getColumnIndexOrThrow( p0 = DBHelper.NIM_COL)
            val mhsNIM = cursor.getString( p0 = cursorNIM)

            txtNama.append("$mhsNama\n")
            txtNIM.append("$mhsNIM\n")
        } while (cursor.moveToNext())
    }
}
```

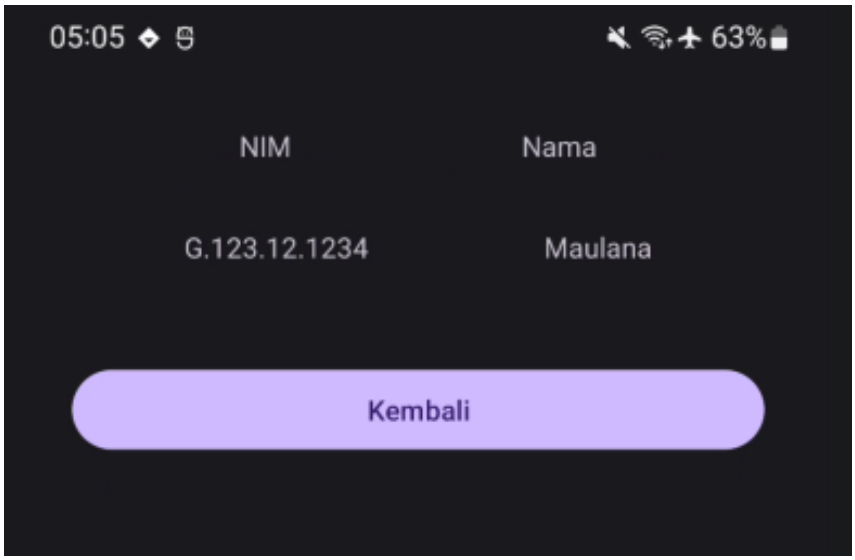
Gambar 7.20: Kode Import TextView

17. Tes aplikasi dengan memasukkan dan menampilkan data



Gambar 7.21: Input Data

18. Klik **Tentang** untuk melihat data



Gambar 7.22: Tampilkan Data

19. Kirim file ke e-learning setelah **Export to ZIP**

Bab 8

Proyek Akhir Aplikasi Android

8.1 Deskripsi Tugas Akhir

Mahasiswa diminta membuat aplikasi mobile berbasis Android menggunakan Android Studio dengan ketentuan berikut:

1. Menggunakan SQLite Database.
2. Implementasi minimal fitur:
    - Create Data
    - Read Data
3. Memiliki minimal 3 Activity, yaitu:
    - Main Activity
    - Daftar/Input Data Activity
    - Lihat Data Activity
4. Aplikasi dapat dijalankan
5. Mengumpulkan:
    - File APK → Kompres ke ZIP
    - File PDF Deskripsi Aplikasi
        (a) Cover
        (b) Nama aplikasi
        (c) Deskripsi aplikasi
        (d) Tujuan aplikasi
        (e) Struktur Activity
        (f) Screenshot aplikasi
        (g) Struktur database SQLite (Tabel biasa)
        (h) Penjelasan fitur Create dan Read

(i) Kesimpulan

(j) Minimal 5 halaman.

8.2 Jenis Aplikasi

8.2.1 Aplikasi Catatan Harian

Fitur:

- Menambah catatan
- Melihat daftar catatan

Tabel SQLite:

- Judul
- Isi catatan
- Tanggal

8.2.2 Aplikasi Daftar Tugas (To-Do List)

Fitur:

- Menambah tugas
- Melihat daftar tugas

Tabel SQLite:

- Nama tugas
- Deadline
- Status

8.2.3 Aplikasi Pendataan Buku

Fitur:

- Menambah data buku
- Melihat daftar buku

Tabel SQLite:

- Judul buku
- Penulis
- Tahun

8.2.4 Aplikasi Data Mahasiswa

Fitur:

- Input data mahasiswa
- Menampilkan data mahasiswa

Tabel SQLite:

- NIM
- Nama
- Program Studi

8.2.5 Aplikasi Inventaris Barang

Fitur:

- Menambah barang
- Melihat stok barang

Tabel SQLite:

- Nama barang
- Jumlah
- Lokasi

8.2.6 Aplikasi Data Kontak Sederhana

Fitur:

- Menambah kontak
- Melihat daftar kontak

Tabel SQLite:

- Nama
- Nomor Telepon
- Email

8.2.7 Aplikasi Pencatatan Pengeluaran

Fitur:

- Menambah pengeluaran
- Melihat riwayat pengeluaran

Tabel SQLite:

- Nama pengeluaran

- Nominal
- Tanggal