Activity dan Intent

Pertemuan 3
Mata Kuliah:
Pemrograman Aplikasi Bergerak

Dosen:
Mahisa Ajy Kusuma, S.Kom., MMSI.



Outline

- Android Project Folder Structure
- Activity
- Intent

Android Project Folder Structure

Android Project Folder Structure

- **Manifests Folder:** Berkas identitas dan aturan utama aplikasi.
- **Java+Kotlin Folder:** Tempat kode logika program disimpan.
- **res (Resources) Folder:** Tempat seluruh aset visual dan konfigurasi UI.
  - Drawable Folder
  - Layout Folder
  - Mipmap Folder
  - Values Folder
- **Gradle Scripts:** Pengatur pustaka (dependencies) dan versi SDK.



Android Manifests

Android Manifest

AndroidManifest.xml adalah berkas konfigurasi utama berformat XML yang wajib ada di setiap proyek aplikasi Android dan bertindak sebagai cetak biru aplikasi yang dibaca pertama kali oleh sistem operasi saat aplikasi diinstal.

Android Manifest juga bertanggung jawab atas pengelolaan hak akses (permissions)—baik izin yang diminta aplikasi dari perangkat (seperti kamera, lokasi, atau penyimpanan) maupun izin yang dibutuhkan aplikasi lain untuk berinteraksi dengannya.



Tentukan nama subclass Anda dengan atribut `name`, menggunakan penetapan paket lengkap. Misalnya, subclass `Activity` dideklarasikan sebagai berikut:

```
<manifest ... >
    <application ... >
        <activity android:name="com.example.myapp.MainActivity" ... >
        </activity>
    </application>
</manifest>
```

Namun, jika karakter pertama dalam nilai `name` adalah titik, namespace aplikasi, dari properti `namespace` milik file `build.gradle` level modul, akan diletakkan sebelum nama. Misalnya, jika namespace adalah `"com.example.myapp"`, nama aktivitas berikut akan me-resolve ke `com.example.myapp.MainActivity`:

```
<manifest ... >
    <application ... >
        <activity android:name=".MainActivity" ... >
            ...
        </activity>
    </application>
</manifest>
```

Permissions

Aplikasi Android harus meminta izin untuk mengakses data pengguna yang sensitif, seperti kontak dan SMS, atau fitur sistem tertentu, seperti kamera dan akses internet. Setiap izin diidentifikasi melalui label yang unik. Misalnya, sebuah aplikasi yang perlu mengirimkan pesan SMS harus memiliki baris berikut dalam manifesnya:

```
<manifest ... >
    <uses-permission android:name="android.permission.SEND_SMS"/>
    ...
</manifest>
```

Mulai dari Android 6.0 (API level 23), pengguna dapat menyetujui atau menolak beberapa izin aplikasi selama runtime. Tidak peduli versi Android mana yang didukung aplikasi, Anda harus mendeklarasikan seluruh permintaan izin dengan elemen `<uses-permission>` dalam manifesnya. Jika izin diberikan, aplikasi tersebut dapat menggunakan fitur yang dilindungi. Jika tidak, upaya aplikasi untuk mengakses fitur tersebut akan gagal.

Aplikasi dapat juga melindungi komponennya sendiri dengan izin. Aplikasi tersebut dapat memiliki semua izin yang ditentukan oleh Android, seperti yang dicantumkan dalam `android.Manifest.permission`, atau izin yang dideklarasikan dalam aplikasi lain. Aplikasi Anda juga dapat menentukan izinnya sendiri. Izin baru akan dideklarasikan dengan elemen `<permission>`.

Untuk mengetahui informasi selengkapnya, lihat Izin di Android.

Device compatibility

File manifes juga merupakan tempat Anda dapat mendeklarasikan tipe fitur hardware atau software yang dibutuhkan aplikasi Anda, pun demikian, tipe perangkat yang kompatibel dengan aplikasi tersebut. Google Play Store tidak mengizinkan pengguna menginstal aplikasi di perangkat yang tidak menyediakan fitur atau versi sistem yang dibutuhkan aplikasi.

Ada beberapa tag manifes yang menentukan perangkat yang kompatibel dengan aplikasi Anda. Berikut ini adalah beberapa tag yang paling umum.

<uses-feature>

Elemen `<uses-feature>` memungkinkan Anda mendeklarasikan fitur hardware dan software yang dibutuhkan aplikasi Anda. Misalnya, jika aplikasi Anda tidak dapat mencapai fungsi dasar di perangkat tanpa sensor kompas, Anda dapat mendeklarasikan sensor kompas seperti yang diperlukan bersama tag manifes berikut:

```
<manifest ... >
    <uses-feature android:name="android.hardware.sensor.compass"
                  android:required="true" />
    ...
</manifest>
```

★ **Catatan**: Jika Anda ingin menyediakan aplikasi Anda di Chromebook, ada beberapa batasan fitur hardware dan software penting yang perlu dipertimbangkan. Untuk mengetahui informasi selengkapnya, lihat Kompatibilitas manifes aplikasi untuk Chromebooks.

<uses-sdk>

Setiap urutan versi platform seringkali menambahkan API baru yang tidak tersedia di versi sebelumnya. Untuk menunjukkan versi minimum dengan perangkat yang kompatibel dengan aplikasi, manifes Anda harus menyertakan tag `<uses-sdk>` dan atribut `minSdkVersion`-nya .

Namun, perhatikan bahwa atribut dalam elemen `<uses-sdk>` digantikan oleh properti yang sesuai dalam file `build.gradle`. Jadi, jika Anda menggunakan Android Studio, tentukan nilai `minSdkVersion` dan `targetSdkVersion` di sini:

Groovy | Kotlin

```
android {
    defaultConfig {
        applicationId = "com.example.myapp"

        // Defines the minimum API level required to run the app.
        minSdkVersion(23)

        // Specifies the API level used to test the app.
        targetSdkVersion(36)
        ...
    }
}
```

Untuk informasi selengkapnya tentang file `build.gradle`, baca tentang cara mengonfigurasi build Anda.

Untuk mengetahui selengkapnya tentang cara mendeklarasikan dukungan aplikasi Anda untuk perangkat yang berbeda, lihat Ringkasan kompatibilitas perangkat.

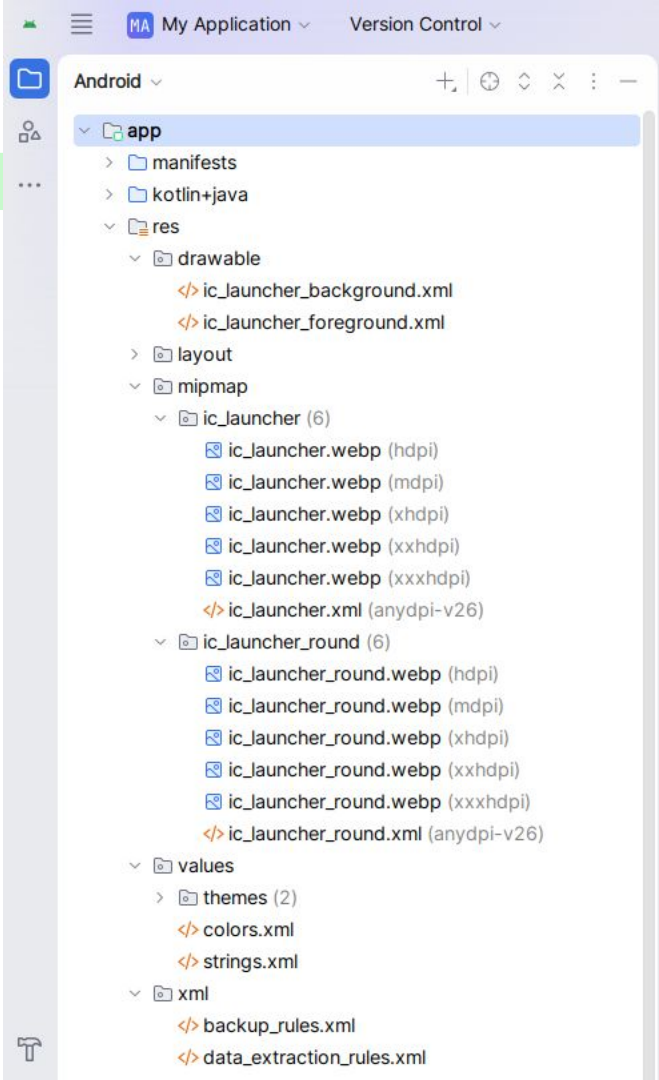
Resource

Resource

Folder res (Resources) adalah direktori khusus dalam proyek Android yang berfungsi memisahkan seluruh aset non-kode dari logika pemrograman Kotlin atau Java.

Pemisahan ini memungkinkan pengembang mengelola tampilan dan perilaku aplikasi secara fleksibel tanpa perlu mengubah kode utama,

sekaligus memudahkan penyesuaian aplikasi terhadap berbagai konfigurasi perangkat, seperti ukuran layar, orientasi, maupun bahasa pengguna



Drawable

Subfolder drawable berfungsi sebagai tempat penyimpanan seluruh aset visual dan grafis yang digunakan dalam antarmuka aplikasi, seperti gambar berformat PNG/JPEG, berkas vektor XML (VectorDrawable), serta bentuk-bentuk kustom seperti button background dan gradient. Pemisahan grafis ke dalam folder ini memudahkan pengembang dalam mengelola tingkat resolusi gambar (density) yang berbeda agar tampilan visual aplikasi tetap tajam dan responsif di berbagai ukuran layar perangkat.

Layout

Subfolder layout berisi berkas-berkas XML yang mendefinisikan struktur arsitektur dan tata letak antarmuka pengguna (user interface). Di dalam folder ini, pengembang menyusun hierarki elemen visual seperti tombol, teks, dan gambar menggunakan komponen View dan ViewGroup, sehingga pemisahan antara desain antarmuka dan logika pemrograman (Kotlin/Java) dapat terwujud secara rapi dan terstruktur.

Mipmap

Subfolder mipmap disediakan khusus oleh Android untuk menyimpan berkas ikon aplikasi (launcher icon) dalam berbagai resolusi, mulai dari mdpi hingga xxxhdpi. Penggunaan folder terpisah ini bertujuan agar peluncur sistem (system launcher) dapat terus menampilkan ikon aplikasi dengan ketajaman terbaik dan rasio yang presisi, tanpa terpengaruh oleh proses optimasi atau pemangkasan resource (resource stripping) berdasarkan resolusi layar perangkat.

Values

Subfolder values digunakan untuk mengelola data-data konstanta non-grafis yang sering digunakan kembali di seluruh bagian aplikasi, seperti teks (strings.xml), warna (colors.xml), dan gaya visual (styles.xml / themes.xml). Pengelompokan ini mempermudah proses pemeliharaan kode, konsistensi desain, serta penerapan fitur lokalisasi (multi-bahasa) dan mode gelap (dark mode) tanpa perlu mengubah logika aplikasi.

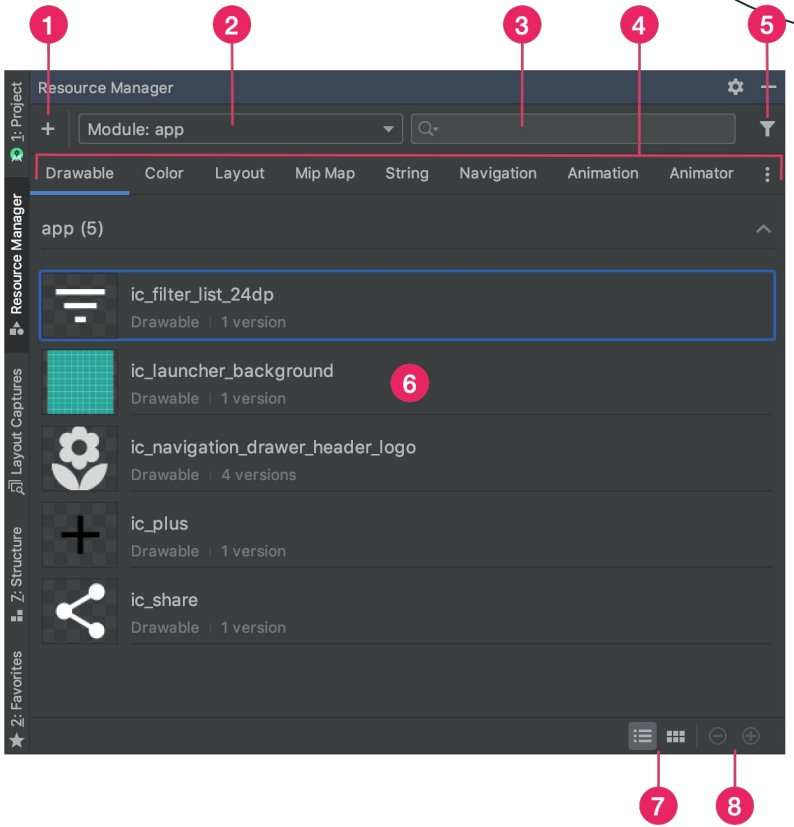
Xml

Subfolder xml dimanfaatkan untuk menyimpan berkas konfigurasi tambahan berformat XML yang bersifat arbitrary atau khusus, yang dapat dibaca oleh aplikasi pada saat runtime. Contoh penggunaan utamanya meliputi konfigurasi skema navigasi (Navigation Graph), aturan preferensi pengguna (Preferences), batas keamanan jaringan (Network Security Configuration), serta pengaturan pustaka pihak ketiga.

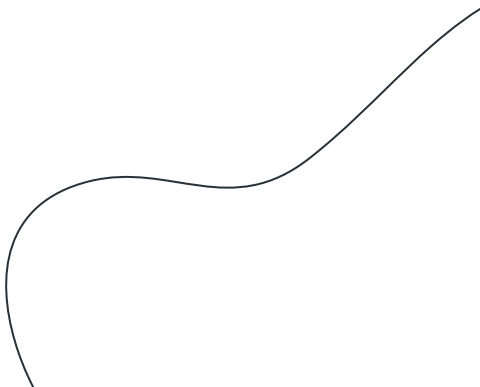
Resource Manager

Android Resource Manager adalah alat (tool) terintegrasi di dalam Android Studio yang berfungsi untuk mengimpor, mengorganisasi, mengolah, dan mempratinjau seluruh aset non-kode (seperti gambar, warna, tata letak, dan font) yang tersimpan di dalam direktori res.

Tool ini memberikan antarmuka visual yang memudahkan pengembang mengelola resource aplikasi secara terpusat tanpa harus membuka folder satu per satu.



Gradle



Apa itu Gradle?

- **Builds automation system**
- **Mengelola siklus build melalui serangkaian task / tugas** (sebagai contoh: mengkompilasi kode sumber Kotlin, menjalankan pengujian/test, dan menginstal aplikasi ke perangkat).
- **Menentukan urutan task yang tepat untuk dijalankan.**
- **Mengelola dependensi** antar proyek maupun pustaka (library) pihak ketiga.



Gradle build file

- **Declare plugins:** Mengaktifkan plugin yang dibutuhkan oleh modul aplikasi
- **Define Android properties:** Meningkatkan/mengatur konfigurasi khusus Android seperti compileSdk, applicationId, minSdk, targetSdk, versionCode, dan versionName.
- **Handle dependencies:** Menentukan library pihak ketiga atau modul internal yang digunakan oleh aplikasi ini.
- **Connect to repositories:** Menentukan lokasi/server mana Gradle harus mengunduh library tersebut

```
plugins {
    alias(libs.plugins.android.application)
}
```

```
dependencies {
    implementation(libs.androidx.activity.ktx)
    implementation(libs.androidx.appcompat)
    implementation(libs.androidx.constraintlayout)
    implementation(libs.androidx.core.ktx)
    implementation(libs.material)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(libs.androidx.junit)
}
```

```
android {
    namespace = "id.ac.stmi.mahisaajy"
    compileSdk {
        version = release( version = 37)
    }

    defaultConfig {
        applicationId = "id.ac.stmi.mahisaajy"
        minSdk = 24
        targetSdk = 37
        versionCode = 1
        versionName = "1.0"

        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            optimization {
                enable = false
            }
        }
    }
    compileOptions {
        sourceCompatibility = JavaVersion.VERSION_11
        targetCompatibility = JavaVersion.VERSION_11
    }
}
```

Gradle Properties

File ini berisi variabel konfigurasi untuk mengoptimalkan kinerja proses build, penggunaan memori, serta mengaktifkan fitur-fitur eksperimental Gradle di Android Studio.

`org.gradle.jvmargs` adalah baris pengatur memori dan karakter untuk mesin Java (JVM) yang menjalankan Gradle Daemon.
Perintah `-Xmx2048m` bertugas menjatah kapasitas RAM maksimal sebesar 2 GB khusus untuk proses *build* Gradle,
sementara `-Dfile.encoding=UTF-8` memaksa seluruh pembacaan dan penulisan file menggunakan standar pengkodean UTF-8 agar tidak terjadi *error* karakter saat dikompilasi di berbagai sistem operasi.

gradle.properties

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. For more details, visit
# https://developer.android.com/r/tools/gradle-multi-project-decoupled-projects
# org.gradle.parallel=true
# When enabled, the Configuration Cache allows Gradle to skip the configuration
# phase entirely if nothing that affects the build configuration (such as build scripts)
# has changed. Additionally, Gradle applies performance optimizations to task execution.
org.gradle.configuration-cache=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
```

Gradle Wrapper Properties

File ini berlokasi di dalam folder gradle/wrapper/gradle-wrapper.properties.

Fungsi utamanya adalah menjamin bahwa proyek ini dikompilasi menggunakan versi Gradle yang sama persis di semua komputer, tanpa mengharuskan Anda atau mahasiswa menginstal Gradle secara manual di OS laptop.

gradle-wrapper.properties

```
#Sat Aug 29 09:27:44 WIB 2026
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
distributionSha256Sum=553c78f50dafcd54d65b9a444649057857469edf836431389695608536d6b746
distributionUrl=https\://services.gradle.org/distributions/gradle-9.5.0-bin.zip
networkTimeout=10000
validateDistributionUrl=true
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists

```

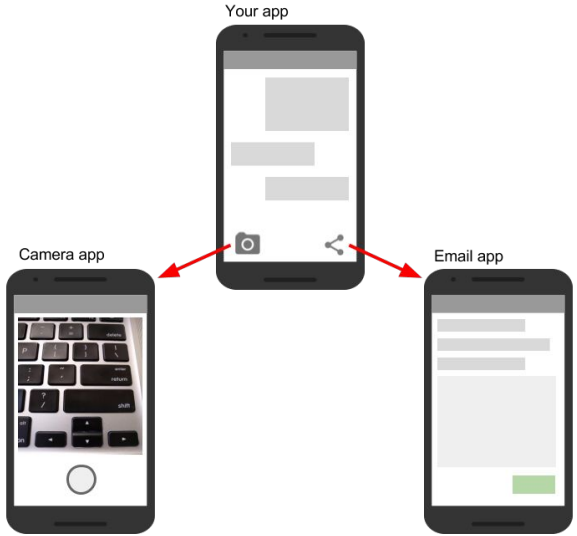
ACTIVITY

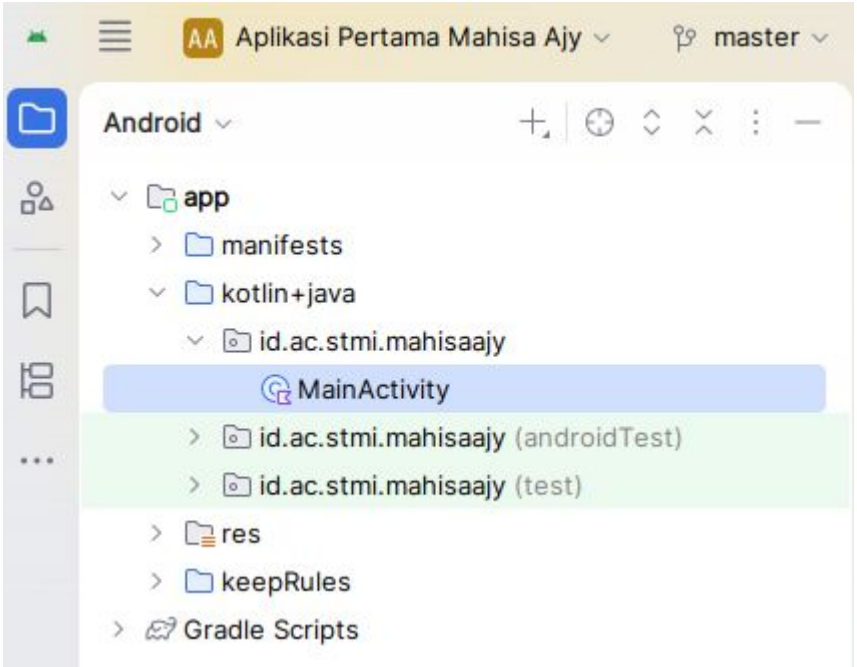
Activity

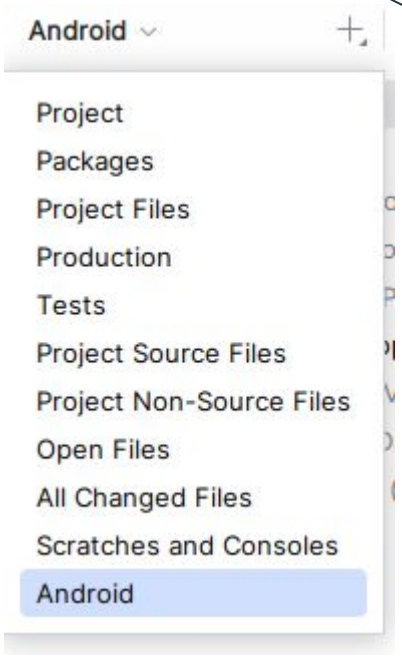
Activity adalah komponen utama dalam arsitektur Android yang menyediakan satu layar antarmuka pengguna (user interface) untuk berinteraksi langsung dengan aplikasi, seperti membaca pesan, mengambil foto, atau melihat peta.

Setiap Activity mengelola siklus hidupnya sendiri (lifecycle) melalui serangkaian metode callback—seperti onCreate(), onStart(), dan onResume()—yang memungkinkan sistem operasi mengatur penggunaan memori dan transisi layar saat pengguna berpindah atau meminimalkan aplikasi.

Dalam struktur proyek, sebuah aplikasi umumnya terdiri dari beberapa Activity yang saling terhubung melalui Intent, di mana setiap Activity wajib didaftarkan ke dalam file AndroidManifest.xml agar dapat dikenali dan dijalankan oleh sistem Android.

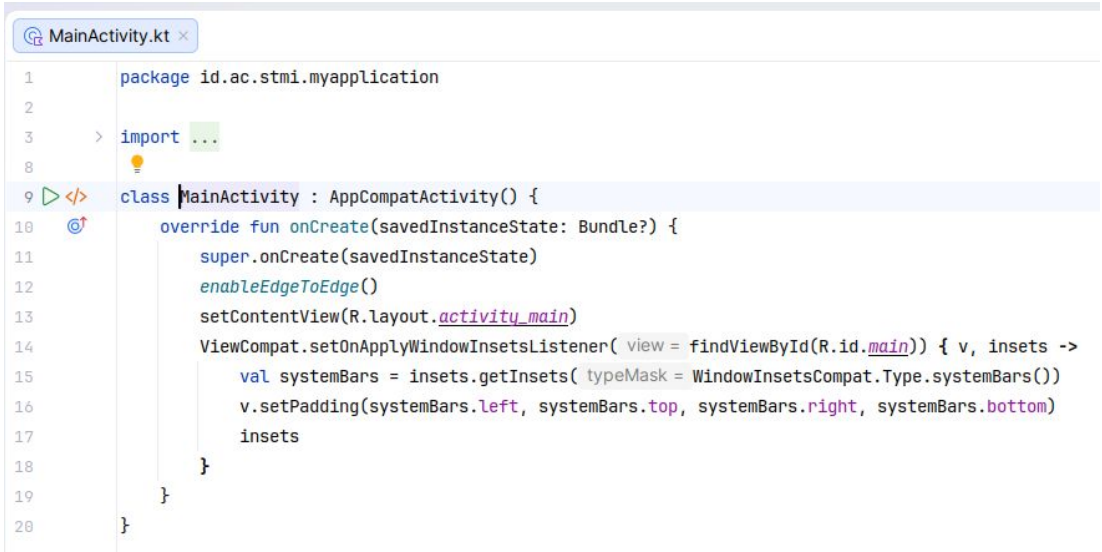



Aplikasi Pertama Mahisa Ajy
master
Android
app
manifests
kotlin+java
id.ac.stmi.mahisaajy
MainActivity
id.ac.stmi.mahisaajy (androidTest)
id.ac.stmi.mahisaajy (test)
res
keepRules
Gradle Scripts


Android
Project
Packages
Project Files
Production
Tests
Project Source Files
Project Non-Source Files
Open Files
All Changed Files
Scratches and Consoles
Android

Main Activity

MainActivity adalah Activity pertama yang dieksekusi dan ditampilkan kepada pengguna saat aplikasi Android dibuka melalui launcher perangkat. Berkas ini bertindak sebagai pintu masuk utama (entry point) aplikasi, yang bertanggung jawab me-render antarmuka layar awal, menginisialisasi dependensi awal, serta mengatur alur navigasi ke komponen atau layar lainnya.



Keberadaan MainActivity didefinisikan secara khusus di dalam AndroidManifest.xml menggunakan kombinasi Intent Filter

<action android:name="android.intent.action.MAIN" />

dan

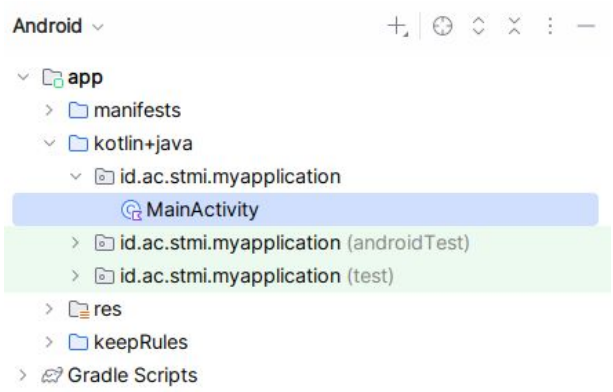
<category android:name="android.intent.category.LAUNCHER" />,

yang memberi tahu sistem operasi Android untuk menampilkan ikon aplikasi di menu utama perangkat dan mengarahkan pengguna ke MainActivity saat ikon tersebut diklik.

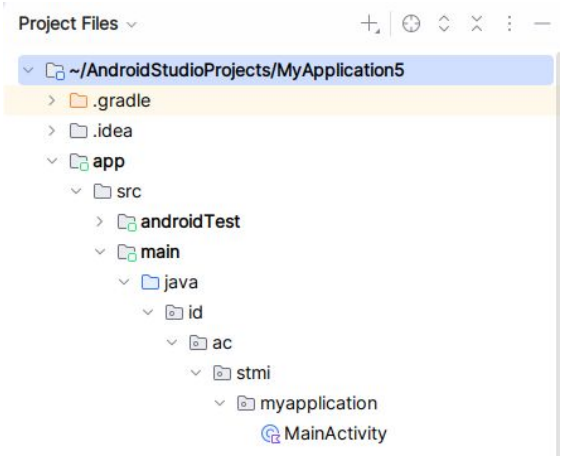
Package

```
package id.ac.stmi.myapplication
```

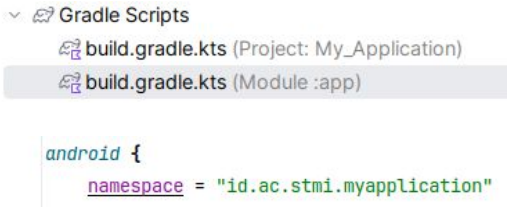
Baris package id.ac.stmi.myapplication adalah deklarasi nama package (paket) dari berkas kode Kotlin tersebut.



Android



Project Files



Nilai package di baris atas file Kotlin/Java kamu harus sama persis dengan nilai namespace yang ada di file build.gradle.kts (Module :app).

Import

```
import android.os.Bundle
```

Fungsi: Mengimpor kelas Bundle milik framework bawaan Android.

```
import androidx.activity.enableEdgeToEdge
```

Fungsi: Mengimpor fungsi ekstensi enableEdgeToEdge() dari pustaka Jetpack AndroidX Activity.

```
import androidx.appcompat.app.AppCompatActivity
```

Kegunaan: Dihubungkan ke kelas kamu melalui pewarisan (class MainActivity : AppCompatActivity()). Kelas ini memberikan dukungan mundur (backward compatibility) agar fitur-fitur antarmuka Android modern tetap dapat berjalan dengan stabil di versi Android lama.

Import

```
import androidx.core.view.ViewCompat
```

Kegunaan: Menyediakan fungsi-fungsi pembantu (helper) untuk memanipulasi elemen UI secara aman lintas versi Android. Di kode kamu, ini digunakan untuk memanggil ViewCompat.setOnApplyWindowInsetsListener(...).

```
import androidx.core.view.WindowInsetsCompat
```

Kegunaan: Digunakan untuk mengambil informasi ukuran area terlarang layar (insets)—seperti tinggi status bar, navigation bar, atau notch. Kode WindowInsetsCompat.Type.systemBars() memanfaatkan kelas ini untuk menghitung safe area agar UI tidak tertutup elemen sistem.

Class MainActivity

```
class MainActivity : AppCompatActivity() {

   override fun onCreate(savedInstanceState: Bundle?) {

       super.onCreate(savedInstanceState)
```

class MainActivity : AppCompatActivity(): Mengindikasikan bahwa MainActivity adalah sebuah komponen halaman (Activity) yang mewarisi fitur-fitur kompatibilitas dari AppCompatActivity.

override fun onCreate(...): Metode siklus hidup (lifecycle) utama yang dipanggil pertama kali oleh sistem Android saat halaman ini dibuat.

super.onCreate(...): Menjalankan proses inisialisasi dasar dari kelas induk (AppCompatActivity).

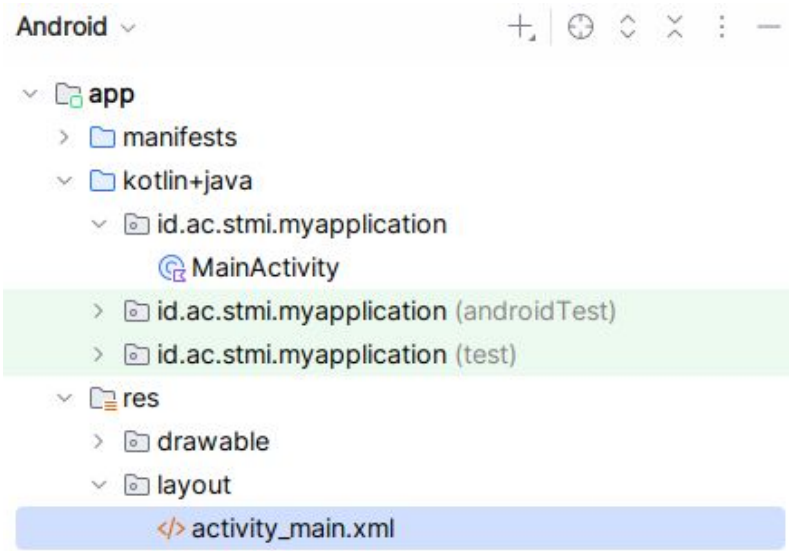
`enableEdgeToEdge()`

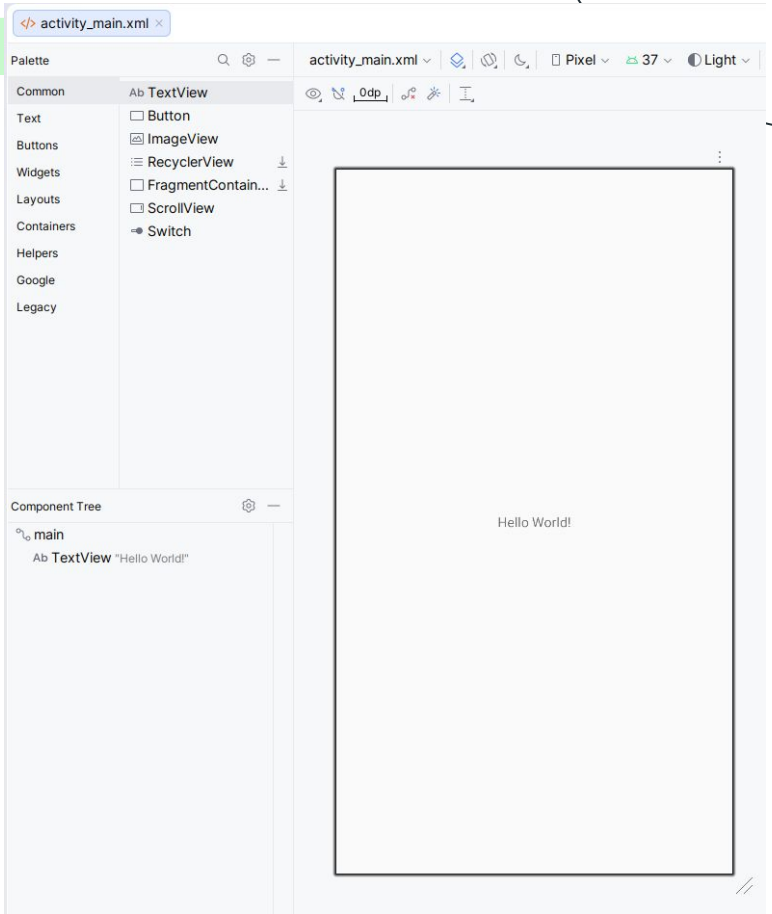
Perintah ini menginstruksikan sistem Android untuk menggambar latar belakang aplikasi hingga memenuhi seluruh area layar—termasuk area di balik Status Bar (atas) dan Navigation/Gesture Bar (bawah).

Mengaktifkan mode Edge-to-Edge, sehingga konten aplikasi dirender secara penuh hingga ke balik status bar (bagian atas jam/baterai) dan navigation bar (tombol/gesture navigasi bawah).

```
setContentView(R.layout.activity_main)
```

Memasang berkas tampilan antarmuka activity_main.xml ke dalam MainActivity agar elemen-elemen UI (teks, tombol, gambar) dirender ke layar.





```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
    Insets
}
```

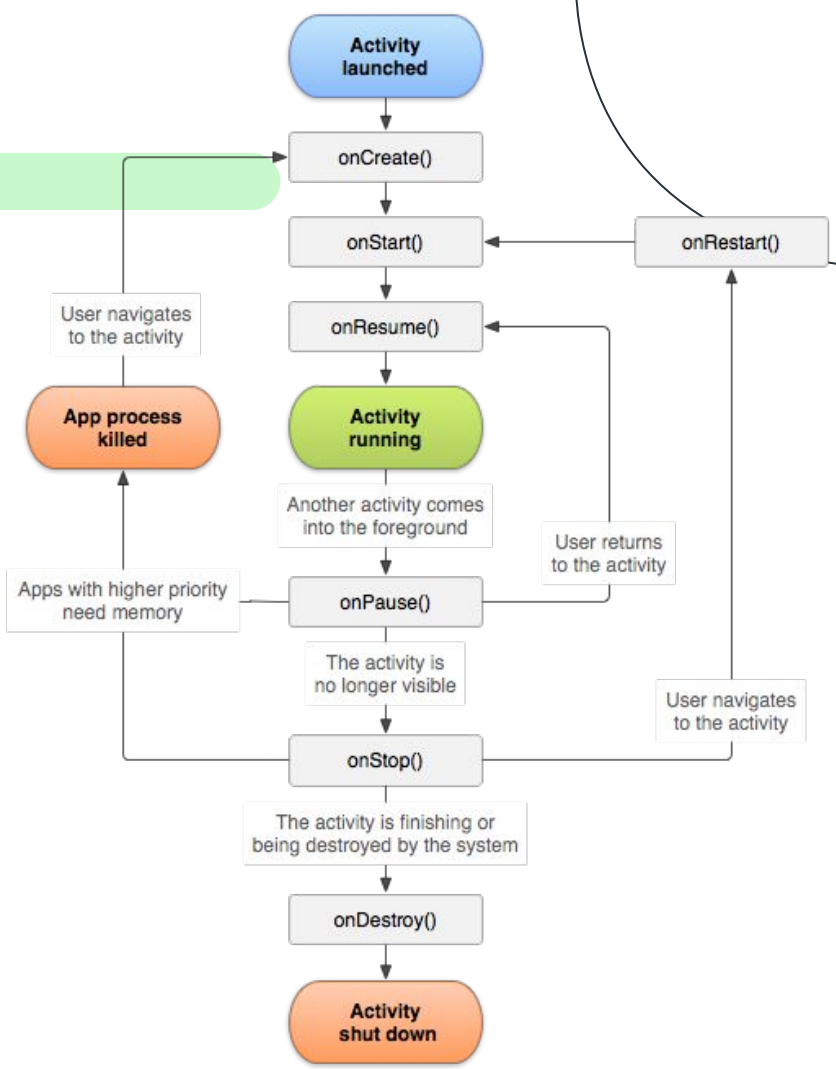
Kode ini berfungsi untuk menghitung dan menyesuaikan jarak aman (safe area) pada layout aplikasi kamu agar tampilannya tidak tertutup oleh elemen bawaan sistem Android (seperti status bar di atas atau navigation/gesture bar di bawah).

- findViewById(R.id.main): Mencari komponen root layout paling luar dari XML activity_main.xml.
- setOnApplyWindowInsetsListener: Memasang pemantau untuk mendeteksi ukuran elemen sistem di layar.
- insets.getInsets(...): Mengambil ukuran tinggi Status Bar dan Navigation Bar dalam satuan piksel secara rinci.
- v.setPadding(...): Memberikan jarak dalam (padding) secara otomatis ke komponen R.id.main sesuai ukuran system bar tersebut, sehingga isi konten tetap berada di area aman.
- insets: Meneruskan data insets ke komponen anak (child view) jika diperlukan.

Activity Lifecycle

Activity

- Untuk menavigasi transisi di antara tahap siklus proses aktivitas, class Activity menyediakan set inti sebanyak enam callback:
  - onCreate,
  - onStart,
  - onResume,
  - onPause,
  - onStop, dan
  - onDestroy.
- Sistem memanggil setiap callback ini saat aktivitas memasuki status baru.



onCreate()

Dipanggil saat Activity pertama kali dibuat. Fungsi ini digunakan untuk menginisialisasi komponen dasar, seperti menetapkan tampilan tata letak (layout XML), mengikat variabel, atau menyiapkan data awal.

onStart()

- Dipanggil ketika Activity mulai terlihat oleh pengguna di layar, namun belum bisa diinteraksi secara penuh. Pada tahap ini, aplikasi bersiap untuk masuk ke tingkat depan (foreground).
- Called after activity:
  - onCreate()
  - or
  - onRestart() if activity was previously stopped

onResume()

Dipanggil saat Activity sudah siap berinteraksi sepenuhnya dengan pengguna. Ini adalah status di mana aplikasi berada di foreground dan menerima semua input pengguna (seperti sentuhan atau ketikan).

onPause()

Dipanggil ketika Activity kehilangan fokus tetapi masih sebagian terlihat di layar (misalnya saat muncul dialog pop-up atau mode split-screen).

Fungsi ini digunakan untuk menghentikan sementara animasi atau operasi yang memakan daya.

onStop()

Dipanggil ketika Activity sudah tidak terlihat sama sekali oleh pengguna karena tertutup oleh layar/aplikasi lain.

Tahap ini digunakan untuk menyederhanakan operasi berat, menyimpan data sementara, atau melepas resource yang tidak diperlukan.

onDestroy()

Dipanggil sebelum Activity benar-benar dihancurkan dan dibersihkan dari memori perangkat—baik karena pengguna menutup aplikasi (menekan tombol back) maupun karena sistem membutuhkan memori tambahan.

onRestart()

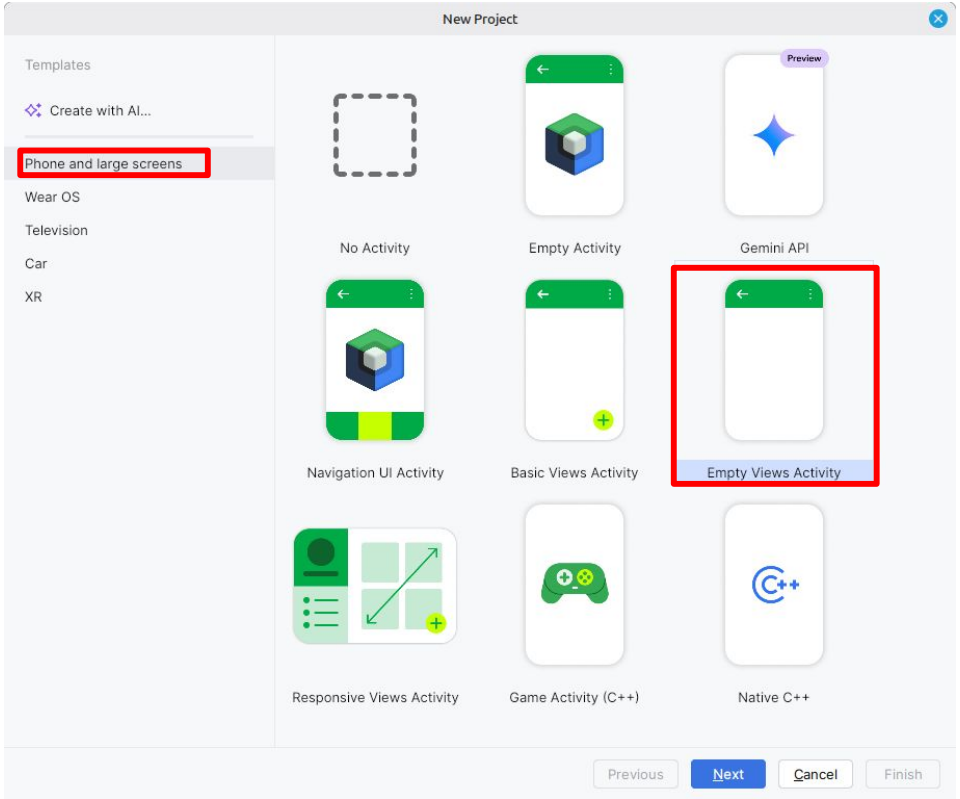
Dipanggil ketika Activity yang tadinya berada dalam kondisi onStop() dibuka kembali oleh pengguna sebelum sempat dihancurkan. Setelah fungsi ini dipanggil, alur akan berlanjut kembali ke onStart().

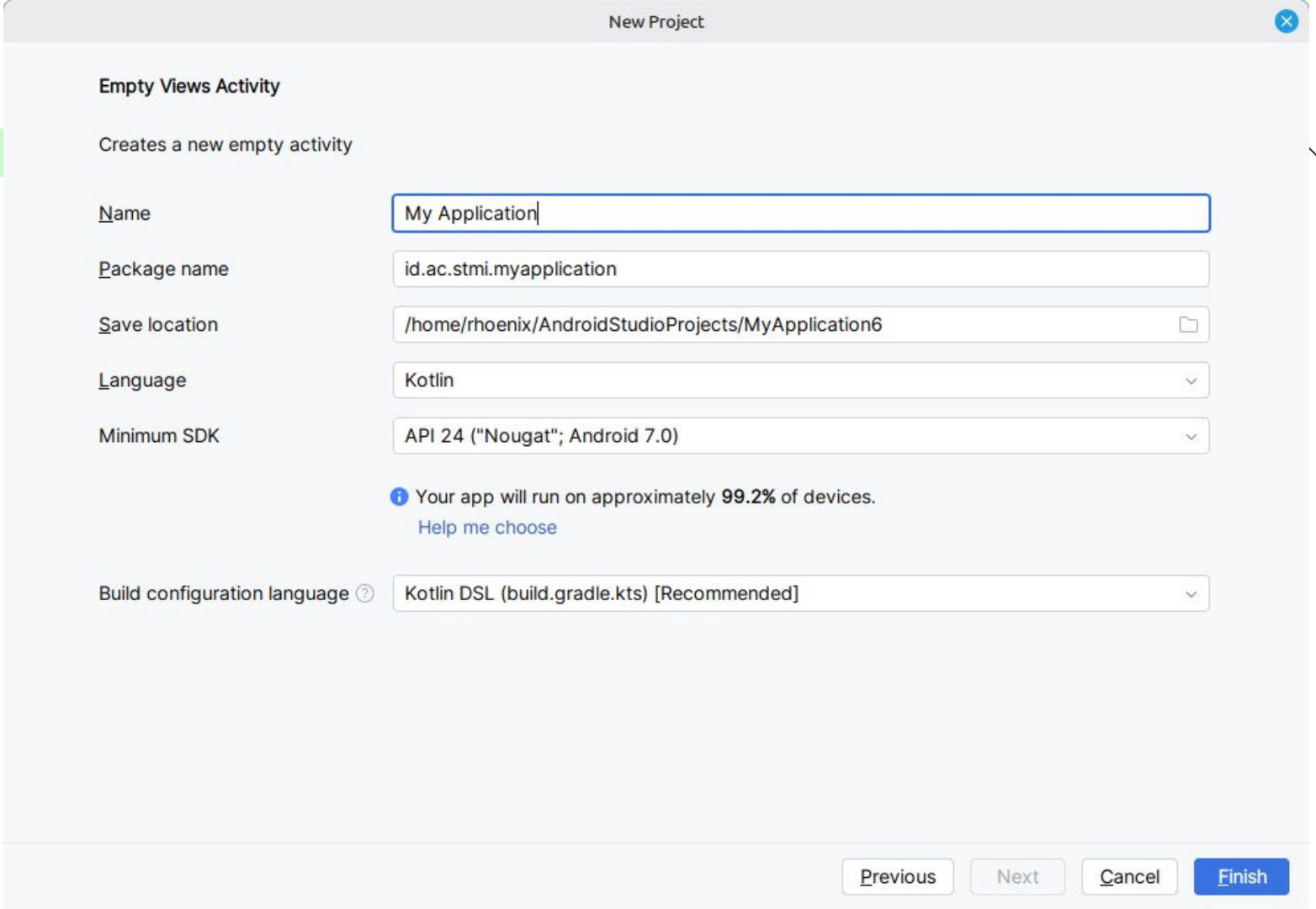
Summary of activity states

State	Callbacks	Description
Created	`onCreate()`	Activity is being initialized.
Started	`onStart()`	Activity is visible to the user.
Resumed	`onResume()`	Activity has input focus.
Paused	`onPause()`	Activity does not have input focus.
Stopped	`onStop()`	Activity is no longer visible.
Destroyed	`onDestroy()`	Activity is destroyed.

Buat Aplikasi

Buat Aplikasi

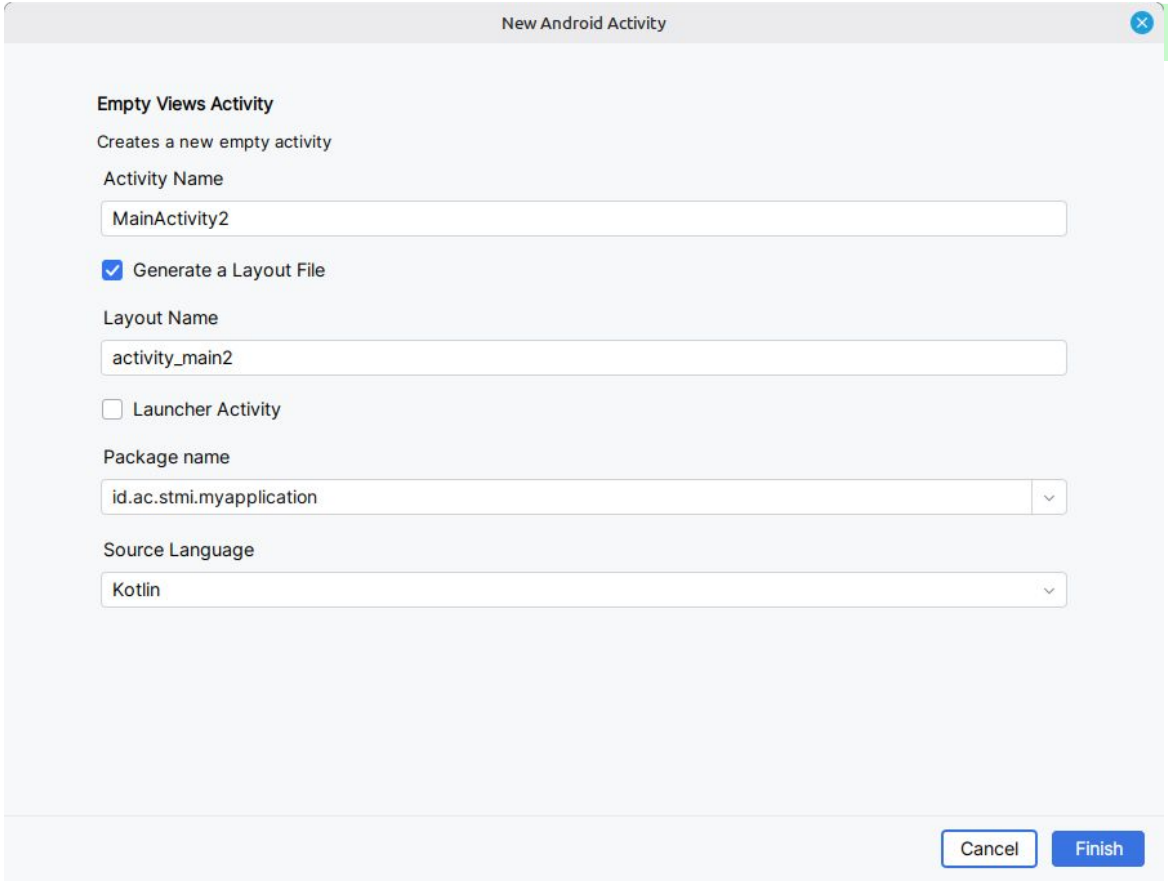


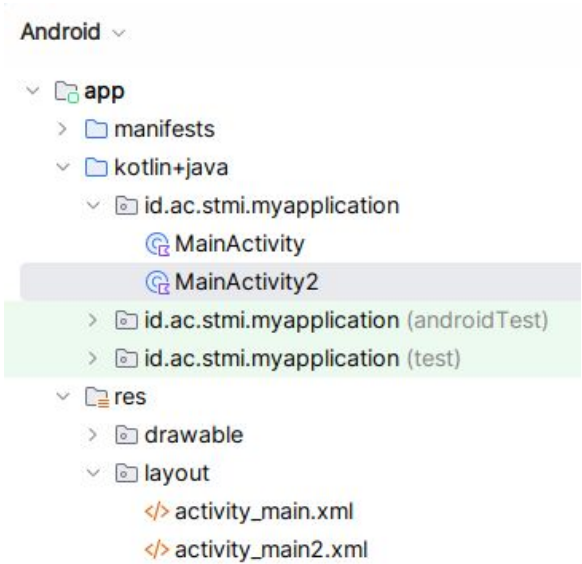

New Project
Empty Views Activity
Creates a new empty activity
Name
My Application
Package name
id.ac.stmi.myapplication
Save location
/home/rhoenix/AndroidStudioProjects/MyApplication6
Language
Kotlin
Minimum SDK
API 24 ("Nougat"; Android 7.0)
Your app will run on approximately 99.2% of devices.
Help me choose
Build configuration language
Kotlin DSL (build.gradle.kts) [Recommended]
Previous
Next
Cancel
Finish

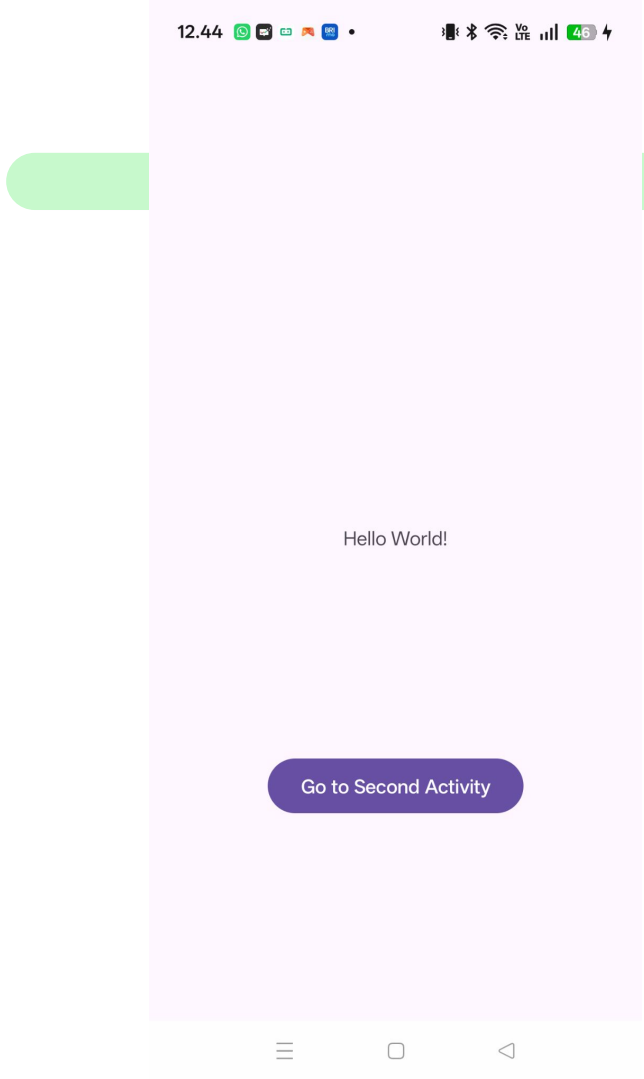
MainActivity.kt

```
11      class MainActivity : AppCompatActivity() {
16          override fun onCreate(savedInstanceState: Bundle?) {
25
26              // 1. Dipanggil saat Activity pertama kali dibuat
27              Toast.makeText( context = this,  text = "onCreate: Activity Dibuat",  duration = Toast.LENGTH_SHORT).show()
28              Log.d(TAG,  msg = "1. onCreate: Activity Dibuat")
29          }
30
31          override fun onStart() {
32              super.onStart()
33              // 2. Dipanggil saat Activity mulai terlihat di layar
34              Toast.makeText( context = this,  text = "onStart: Activity Terlihat",  duration = Toast.LENGTH_SHORT).show()
35              Log.d(TAG,  msg = "2. onStart: Activity Terlihat")
36          }
37
38          override fun onResume() {
39              super.onResume()
40              // 3. Dipanggil saat Activity mendapat fokus dan siap berinteraksi
41              Toast.makeText( context = this,  text = "onResume: Activity Siap Berinteraksi",  duration = Toast.LENGTH_SHORT).show()
42              Log.d(TAG,  msg = "3. onResume: Activity Siap Berinteraksi")
43          }
```

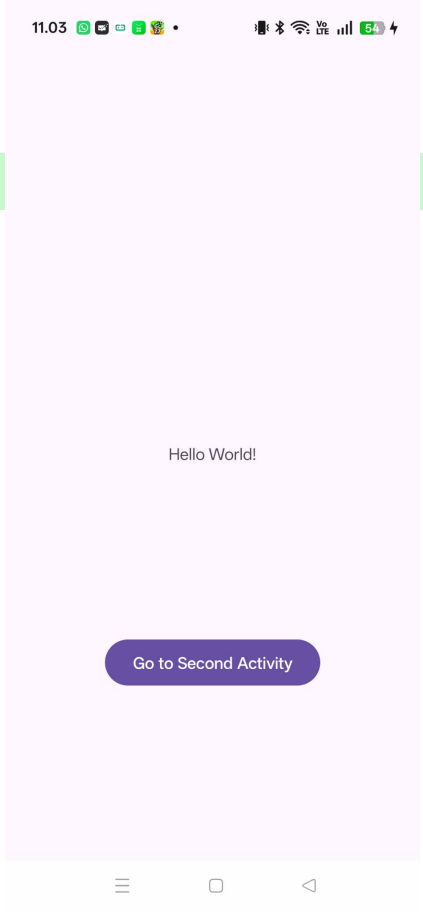
File -> New -> Activity -> Empty Views Activity






12.44
Hello World!
Go to Second Activity


12.45
Ini adalah halaman Second Activity



Activity 1

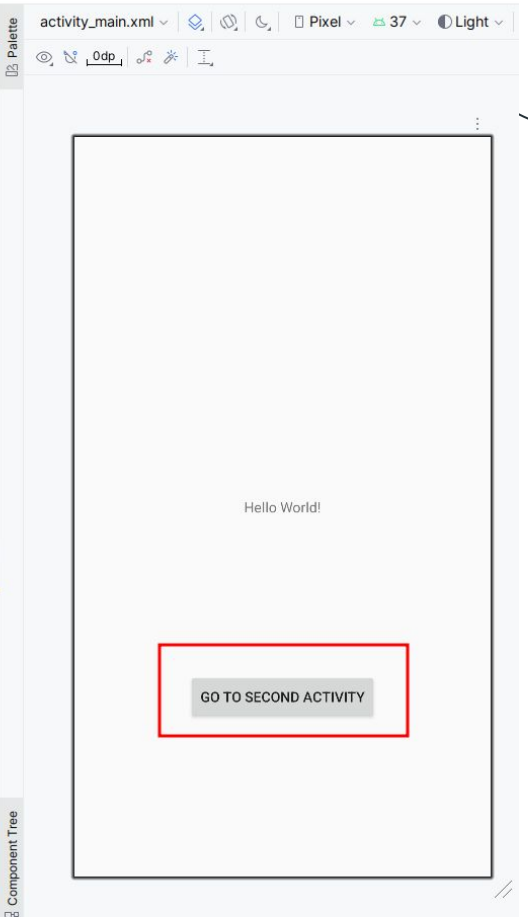
activity_main.xml

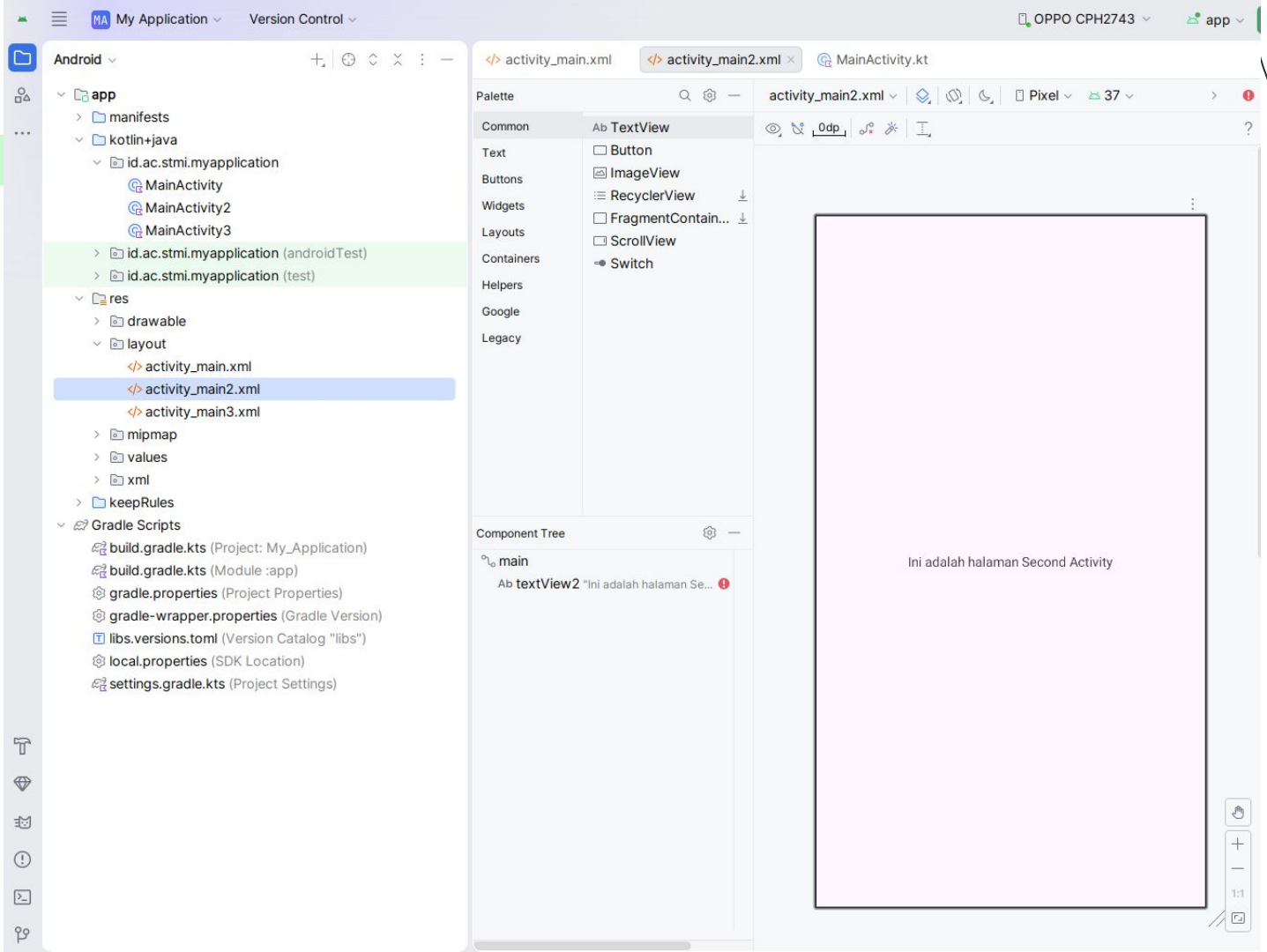
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android=
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second Activity"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

</androidx.constraintlayout.widget.ConstraintLayout>
```




My Application
Version Control
OPPO CPH2743
app
Android
app
manifests
kotlin+java
id.ac.stmi.myapplication
MainActivity
MainActivity2
MainActivity3
id.ac.stmi.myapplication (androidTest)
id.ac.stmi.myapplication (test)
res
drawable
layout
activity_main.xml
activity_main2.xml
activity_main3.xml
mipmap
values
xml
keepRules
Gradle Scripts
build.gradle.kts (Project: My_Application)
build.gradle.kts (Module :app)
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
libs.versions.toml (Version Catalog "libs")
local.properties (SDK Location)
settings.gradle.kts (Project Settings)
activity_main.xml
activity_main2.xml
MainActivity.kt
Palette
Common
Text
Buttons
Widgets
Layouts
Containers
Helpers
Google
Legacy
TextView
Button
ImageView
RecyclerView
FragmentContain...
ScrollView
Switch
Component Tree
main
textView2 "Ini adalah halaman Se...
activity_main2.xml
Pixel
37
0dp
Ini adalah halaman Second Activity
1:1

Intent

Intent adalah mekanisme pesan dalam Android yang berfungsi sebagai jembatan komunikasi dan pemicu aksi (action) antar-komponen aplikasi. Tidak hanya menghubungkan interaksi antar-Activity di dalam satu aplikasi,

Intent juga mampu membawa data serta memicu aplikasi luar—seperti membuka peta di Google Maps atau mengirim email via Gmail saat pengguna menekan suatu tombol.

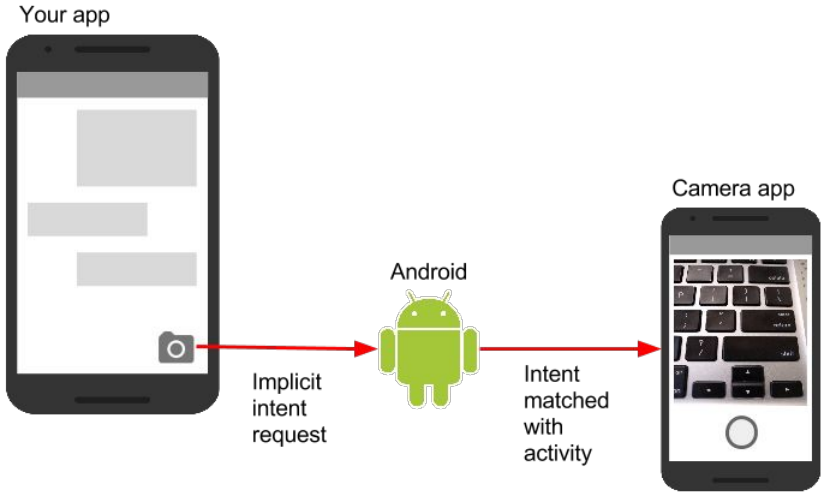
Explicit Intent

Explicit Intent adalah jenis Intent yang menentukan komponen tujuan secara langsung dan spesifik dengan menyebutkan nama kelasnya, biasanya digunakan untuk navigasi internal seperti berpindah dari MainActivity ke DetailActivity di dalam satu aplikasi.

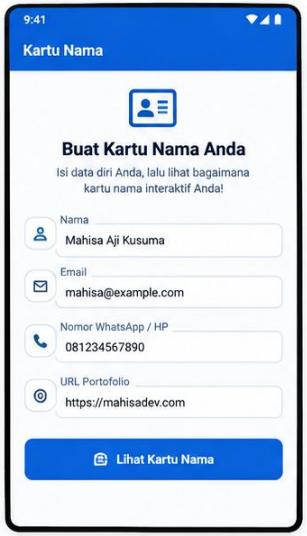


Implicit Intent

Sebaliknya, Implicit Intent tidak menyebutkan nama komponen tujuan secara eksplisit, melainkan mendefinisikan aksi umum yang ingin dilakukan—seperti membuka tautan web (ACTION_VIEW), mengambil foto (ACTION_IMAGE_CAPTURE), atau mengirim pesan—sehingga sistem Android akan mencari dan menampilkan opsi aplikasi yang sesuai di perangkat (seperti Google Maps atau Gmail) untuk menangani permintaan tersebut.



Tugas



Explicit Intent



1. MainActivity (Form Input)

Pengguna mengisi nama, email, nomor WhatsApp/HP, dan URL portofolio, kemudian menekan tombol "Lihat Kartu Nama".

2. ProfileActivity (Tampilan Kartu Nama)

Data yang dikirim dari MainActivity ditampilkan dalam bentuk UI kartu nama dengan 3 tombol aksi.

Tombol Aksi (Implicit Intent)



Kunjungi Website

Membuka URL portofolio di Browser.



Hubungi Saya

Membuka Dial pad HP dengan nomor yang diisi.



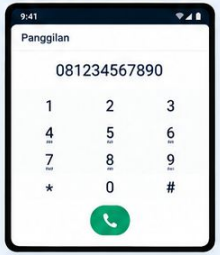
Bagikan Kartu

Mengirim teks rangkuman profil ke aplikasi lain (WhatsApp/Email/Notes).

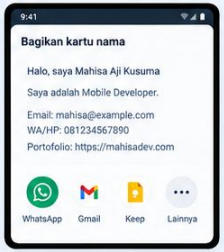
Contoh Implicit Intent - Browser



Contoh Implicit Intent - Dialer



Contoh Implicit Intent - Share



Inti Skenario

1. **Form Input (MainActivity)**
   Input: Nama, Email, WhatsApp/HP, URL Portofolio.
2. **Explicit Intent**
   Kirim data ke ProfileActivity untuk ditampilkan sebagai kartu nama.
3. **Implicit Intent**
   3 tombol aksi:
   - Kunjungi Website (Browser)
   - Hubungi Saya (Dialer)
   - Bagikan Kartu (Share ke aplikasi lain)