DASAR PEMOGRAMAN MOBILE ANDROID DENGAN KOTLIN

Ade Kurniawan
Jusmardi
Widya Darwin
Melri Deswina

DUMMY

UNDANG-UNDANG REPUBLIK INDONESIA
NO 19 TAHUN 2002
TENTANG HAK CIPTA
PASAL 72
KETENTUAN PIDANA SANGSI PELANGGARAN

1. Barang siapa dengan sengaja dan tanpa hak mengumumkan atau memperbanyak suatu Ciptaan atau memberi izin untuk itu, dipidana dengan pidana penjara paling singkat 1 (satu) bulan dan denda paling sedikit Rp 1.000.000, 00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan denda paling banyak Rp 5.000.000.000, 00 (lima milyar rupiah)
2. Barang siapa dengan sengaja menyerahkan, menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu Ciptaan atau barang hasil pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksud dalam ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan denda paling banyak Rp 500.000.000, 00 (lima ratus juta rupiah).

DUMMY

DASAR PEMOGRAMAN MOBILE ANDROID DENGAN KOTLIN

DUMMY

Ade Kurniawan, Jusmardi, Widya Darwin, Melri Deswina

UNP PRESS

2025

DASAR PEMOGRAMAN MOBILE ANDROID DENGAN KOTLIN

editor, Tim editor UNP Press
Penerbit UNP Press, Padang, 2025
1 (satu) jilid; 17.6 x 25 cm (B5)
Jumlah Halaman x + 216 Halaman Buku

ISBN :

DUMMY

DASAR PEMOGRAMAN MOBILE ANDROID DENGAN KOTLIN

Hak Cipta dilindungi oleh undang-undang pada penulis
Hak penerbitan pada UNP Press

Penyusun: Ade Kurniawan, Jusmardi, Widya Darwin, Melri Deswina.
Editor Substansi: Dr. Phil. Dony Novaliendry, M.Kom.
Editor Bahasa: Prof. Dr. Harris Effendi Thahar, M.Pd.
Desain Sampul & Layout: Ridha Prima Adri, Tommy Arjuna Firdaus.

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan Yang Maha Esa atas selesainya buku Dasar Pemrograman Mobile Android ini. Buku ini disusun sebagai panduan bagi pemula yang ingin menguasai dasar-dasar Kotlin dan mengembangkan aplikasi Android menggunakan Android Studio.

Dalam era digital saat ini, kemampuan mengembangkan aplikasi mobile menjadi keterampilan yang penting. Kotlin, sebagai bahasa resmi untuk Android, memberikan efisiensi dalam penulisan kode. Android Studio, dengan fitur lengkapnya, mendukung pengembangan aplikasi secara optimal.

Buku ini disusun bertahap, mulai dari fitur Android Studio, sintaks Kotlin, hingga latihan membangun aplikasi sederhana, sehingga pembaca dapat mengikuti alur pembelajaran dengan mudah. Kami berterima kasih kepada semua pihak yang telah membantu dalam penuntasan buku edisi pertama ini. Selanjutnya untuk edisi mendatang kami terbuka untuk menerima saran dari pembaca semua.

Semoga buku ini bermanfaat dan membantu pembaca mengembangkan keterampilan mengembangkan aplikasi mobile Android dengan Kotlin. Selamat belajar!

Padang, November 2024

Penulis

DUMMY

DAFTAR ISI

DUMMY

DUMMY

DUMMY

DAFTAR GAMBAR

DUMMY

DAFTAR TABEL

DUMMY

BAB 1
PENDAHULUAN

A. Deskripsi Mata Kuliah

Mata kuliah Pemrograman Sistem Bergerak ini merupakan mata kuliah yang berfokus pada pengembangan pengetahuan dan keterampilan mahasiswa dalam merancang, mengimplementasikan, dan mengoptimalkan aplikasi mobile pada platform Android. Mata kuliah ini mencakup dasar-dasar pemrograman Android, arsitektur sistem operasi mobile, serta pengenalan terhadap berbagai alat dan teknik yang digunakan dalam membangun aplikasi yang fungsional, efisien, dan ramah pengguna. Ketik konten Sub Bab disini.

B. Capaian Pembelajaran Lulusan (CPL)

Mahasiswa mampu mengembangkan aplikasi mobile yang fungsional dan efisien, menggunakan platform Android, serta memahami arsitektur sistem operasi mobileKetik konten disini

C. Capaian Pembelajaran Mata Kuliah (CPMK)

CPL-1 (S9)	Bertaqwa kepada Tuhan Yang Maha Esa dan mampu menunjukkan sikap religius.
CPL-2 (S8)	Menginternalisasi nilai, norma, dan etika akademik.
CPL-3 (P)	Menguasai pengetahuan dasar untuk mengembangkan aplikasi sistem bergerak (*mobile*) yang fungsional dan efisien, menggunakan platform Android, serta memahami arsitektur sistem operasi mobile.
CPL-4 (P)	Menguasai konsep dan prinsip-prinsip: perancangan dan pembangunan perangkat lunak dengan metode perencanaan, rekayasa kebutuhan, perancangan, pengimplementasian, pengujian, dan peluncuran yang baku dan ilmiah, dan menghasilkan produk perangkat lunak yang memenuhi berbagai parameter kualitas
CPL-5 (KU2)	Mampu menunjukkan kinerja mandiri, bermutu, dan terukur
CPL-6 (KK)	Mampu merancang dan menerapkan komponen

DUMMY

utama dalam arsitektur Android seperti *Activity*, *Fragment*, Service, dan Broadcast Receiver, serta bagaimana komponen ini bekerja dalam siklus hidup aplikasi.

D. Asesmen dan Evaluasi Pembelajaran

1. Penilaian Harian: Penilaian ini dilakukan untuk mengetahui pemahaman konsep dasar pemrograman Android, arsitektur aplikasi, dan komponen UI. Penilaian harian ini dilakukan setiap pertemuan di dalam kelas.
2. Proyek Tengah Semester: Proyek aplikasi kecil dengan fitur dasar (seperti antarmuka pengguna dan penyimpanan data) yang melibatkan penggunaan *Activity* dan *Fragment* serta interaksi sederhana dengan database atau API.
3. Ujian Tertulis atau Ujian Praktik: Ujian ini menguji pemahaman teori serta kemampuan praktik mahasiswa terkait dengan struktur aplikasi, siklus hidup Android, dan penggunaan API dasar.
4. Proyek Akhir: Mahasiswa mengembangkan aplikasi mobile yang lebih kompleks, misalnya dengan fitur lengkap seperti penggunaan GPS, integrasi API eksternal, dan optimasi performa.Ketik konten Sub Bab disini.

E. Petunjuk Penggunaan Buku Ajar bagi Dosen dan Mahasiswa

Petunjuk bagi Dosen

1. Buku ajar ini berisi materi teori dan praktek dasar pemrograman mobile android menggunakan bahasa pemrograman Kotlin dan compiler Android Studio. Penuhi minimum prerequisite hardware dan software agar anda dapat menjalankan langkah-langkah pembuatan aplikasi dengan lancar.
2. Rencanakan pembelajaran berdasarkan struktur buku yang mencakup pengenalan pemrograman Android dan komponen dasar sistem pemrograman sistem bergerak (*mobile*) menggunakan bahasa pemrograman Kotlin.
3. Setiap materi memiliki pembahasan teori di awal bab yang diharapkan untuk dipahami oleh mahasiswa terlebih dahulu. Sampaikan pengantar teori ini untuk mengkonstruksi pemahaman mahasiswa dan sebagai pengantar untuk kegiatan praktek.

DUMMY

4. Buku ini mencakup panduan penggunaan alat-alat seperti Android Studio, bahasa pemrograman Kotlin, dan alat pengujian. Pastikan mahasiswa mengikuti langkah-langkah ini, dan berikan bimbingan tambahan jika mereka mengalami kesulitan:
5. Tugaskan latihan praktik sesuai dengan materi praktek. Buku ini dilengkapi dengan contoh kode, dan langkah-langkah implementasi program. Gunakan ini sebagai referensi untuk menyelesaikan tugas.
6. Manfaatkan buku ajar sebagai panduan teori bagi proyek atau tugas akhir mahasiswa. Instruksikan mahasiswa untuk merujuk ke bab-bab yang sesuai ketika mereka mengerjakan proyek.

Petunjuk bagi Mahasiswa

1. Mulailah dengan membaca bagian teori untuk memahami dasar-dasar pemrograman sistem bergerak, termasuk arsitektur Android dan siklus hidup aplikasi.
2. Buku ajar ini berisi contoh langkah demi langkah yang membantu anda memahami konsep-konsep penting. Ikuti contoh ini dengan seksama dan praktikkan sendiri di Android Studio. Saat menggunakan buku jalankan kode di perangkat Anda untuk benar-benar memahami cara kerjanya.
3. Gunakan buku sebagai referensi selama mengerjakan tugas atau proyek akhir. Buka kembali bab-bab yang relevan di buku ajar. Gunakan buku sebagai referensi untuk menemukan solusi atau inspirasi dalam menyelesaikan permasalahan pemrograman sistem bergerak.
4. Diskusikan materi dalam kelompok atau dengan dosen jika menemukan kendala yang sulit untuk diselesaikan.

DUMMY

F. Rencana Program Semester

Table 1. RPS Mata Kuliah

RPS MATA KULIAH					
Mata Kuliah	**Kode MK**	**RumpunMK**	**Bobot (SKS)**	**Semester**	**Tanggal Penyusunan**
Praktikum Pemrograman SistemBergerak	INF1.62.5007	Wajib-Prodi	3 SKS	5/8	22 Agustus 2024
	Dosen Pengembang RPS		**Koordinator Prodi Informatika**		**Kepala Departemen**

Tim Pengembang Mata Kuliah Prodi Informatika Jurusan Teknik ElektronikaFT UNP	**Ade Kurniawan, S.Pd., M.Pd.T** **NIDN. 0029069202**	**Dr. Yeka Hendriyani, M.Kom** **NIP. 198405202010122003**	**Dr. Hendra Hidayat, M.Pd** **NIP. 198703052020121012**

***Learning Outcomes* (LO)/ Capaian Pembelajaran (CP)**	**Capaian Pembelajaran (CP) Prodi**	
	CPL-1 (S9)	Bertaqwa kepada Tuhan Yang Maha Esa dan mampu menunjukkan sikap religius.
	CPL-2 (S8)	Menginternalisasi nilai, norma, dan etika akademik.
	CPL-3 (P)	Menguasai pengetahuan dasar untuk mengembangkan aplikasi sistem bergerak (*mobile*) yang fungsional dan efisien menggunakan platform Android, serta memahami arsitektur sistem operasi *mobile*.
	CPL-4 (P)	Menguasai konsep dan prinsip-prinsip: perancangan dan pembangunan perangkat lunak dengan metode perencanaan, rekayasa kebutuhan, perancangan, pengimplementasian, pengujian, dan peluncuran yang baku dan ilmiah, dan menghasilkan produk perangkat lunak yang memenuhi berbagai parameter kualitas secara teknis maupun manajerial, dan berdaya guna serta menguasai konsep dan prinsip-prinsip: pembuatan program sederhana dalam bahasa pemrograman umum maupun bahasa pemrograman berorientasi objek, pembuatan aplikasi web dan aplikasi desktop, pembuatan basisdata sederhana untuk menyelesaikan permasalahan dalam konteks pengembangan perangkat lunak secara umum.
	CPL-5 (KU2)	Mampu menunjukkan kinerja mandiri, bermutu, dan terukur
	CPL-6 (KK)	Mampu merancang dan menerapkan komponen utama dalam arsitektur Android seperti *Activity*, *Fragment*, Service, dan Broadcast Receiver, serta bagaimana komponen ini bekerja dalam siklus hidup aplikasi.
	Capaian Pembelajaran Mata Kuliah (CP-MK)	
	CPMK-1	Mahasiswa mampu menjelaskan konsep dasar pemrograman sistem bergerak, konsep fundamental dalam pemrograman sistem bergerak berbasis Andorid dan melakukan instalasi software untuk membuat aplikasi Android dengan Kotlin.
	CPMK-2	Menjelaskan konsep *Activity* dan *Permission*, *feature* serta dapat mengimplementasikan *Activity*.
	CPMK-3	Menjelaskan konsep intent dan mengimplementasikan intent eksplisit dan inten implisit.
	CPMK-4	Menjelaskan konsep *Fragment* dan alur hidup *Fragment* dan hubungan *Fragment* dengan *Activity*
CP	CPMK-5	Membandingkan konseplayout dan mengimplementasikan layout
	CPMK-6	Menjelaskan SearchView, NestedScrollView di android dan mengimplementasikannya
	CPMK-7	Menjelaskan konsep *Adapter* dan *RecyclerView*, serta mengimplementasikan *RecyclerView* di Android
	CPMK-8	Membandingkan penggunaan dialog, toast, snackbar dan actionbar serta menerapkannya pada aplikasi Android
	CPMK-9	Menjelaskan notifikasi di android dan mengimplementasikan notifikasi
	CPMK-10	Menjelaskan dan mengimplementasikan Sharedpreference
	CPMK-11	Menganalisis Web Service di android dan mengimplementasikan HTTP Request dengan Retrofit

Deskripsi Mata Kuliah	Mata kuliah Pemrograman Sistem Bergerak ini merupakan mata kuliah yang berfokus pada pengembangan pengetahuan dan keterampilan mahasiswa dalam merancang, mengimplementasikan, dan mengoptimalkan aplikasi *mobile* pada platform Android. Mata kuliah ini mencakup dasar-dasar pemrograman Android, arsitektur sistem operasi mobile, serta pengenalan terhadap berbagai alat dan teknik yang digunakan dalam membangun aplikasi yang fungsional, efisien, dan ramah pengguna.
Pustaka	**Pustaka Utama** **Bahan Bacaan Utama (BU)** [1] Josh Skeen, David Greenhalgh. Kotlin Programming: The Big Nerd Ranch Guide. Packt Publishing.2019. [2] John Horton, Android Programming with Kotlin for Beginners. Packt Publishing .2019. [3] Dawn Griffiths, David Griffiths. Head First Android Development: A Brain-Friendly Guide (3rd Edition). O'Reilly Media.2021 **Pustaka Pendukung** **Bahan Bacaan Pendukung (BP)** [1] Mandar Agashe, Modern Android App Development with Jetpack and Kotlin. Apress.2021 [2] Wahana Komputer, Pemrograman Aplikasi Android dengan Kotlin. Andi Publisher. 2018

	Perangkat Lunak	**Perangkat Keras**
Media Pembelajaran	1. elearning.unp.ac.id 2. Modul Ajar 3. Android Studio 4. JDK dan SDK Android	1. Laptop/Komputer PC/Notebook 2. Handphone/Tablet 3. Layar dan LCD Proyektor
Mata Kuliah Prasyarat	-	

DUMMY

Rencana Kegiatan Pembelajaran:

Table 2. Rencana Kegiatan Pembelajaran

Minggu ke-	Sub-CPMK (Kemampuan Akhiryang diharapkan)	Materi Pembelajaran	Metode / Strategi Pembelajaran	Alokasi Waktu	Aktifitas Pembelajaran	Kriteria /Indikator Penilaian	Referensi	Bobot (%)
1	2	3	4	5	6	7	8	9
1-2	Mahasiswa dapat menjelaskan konsep dasar pemrograman sistem bergerak dan melakukan instalasi android studio	• Konsep dasar Pemrograman Sistem Bergerak • Komponen Android • Alur hidup aplikasi Andorid • Instalasi Android Studio • Konfigurasi SDK • Membuat program hello world	• Ceramah • Demonstrasi • Penugasan	3 x 50 menit	Dosen memberikan materi konsep dasar Pemrograman Sistem Bergerak, komponen Android dan alur hidup aplikasi android. Mahasiswa Mengenali konsep dasar Kotlin dan bagaimana mengintegrasikannya dengan Android Studio. Mahasiswa mampu menulis aplikasi sederhana pertama menggunakan Kotlin dan Android Studio.	• Ketepatan dalam menjelasakan konsep dasar pemrograman sistem bergerak dan implementasi program hello world	BU-1,2 BP-1	5%
3	Mahasiswa dapat menjelaskan konsep *Activity* dan *Permission, feature* serta dapat mengimplementasikan penggunaan *Activity*	• Alur hidup *Activity* • Contoh *Activity* • Alur hidup *Permission*	• Ceramah • Tanya Jawab • Latihan • Project base learning	3 x 50 menit	Dosen memberikan materi dan kemudian memberikan beberapa contoh penggunaan *Activity* dan alur hidup *Activity*. Mahasiswa Mahasiswa melakukan implementasi *Activity* menggunakan Kotlin.	• Kemampuan mahasiswa dalam mengimplementasikan	BU-1,2 BP-1	5%
4	Mahasiswa mampu menjelaskan dan mengimplementasikan intent	• Intent • Intent Explicit • Intent Implicit	• Demonstrasi • Project based learning • Laporan	3x50 menit	Dosen menjelaskan tentang intent dan kemudian memberikan contoh program yang memerlukan penggunaan intent. Mahasiswa memahami dan mengimplementasikan penggunaan intent	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,2 BP-1	5%
5	Mahasiswa mampu menjelaskan dan mengimplementasikan penggunaan *Fragment*	• *Fragment* • Alur hidup *Fragment*	• Demonstrasi • Tanya Jawab • Project based learning • Laporan	3x50 menit	Dosen menjelaskan tentang *Fragment* dan kemudian memberikan contoh program yang memerlukan penggunaan *Fragment.* Mahasiswa memahami dan mengimplementasikan penggunaan *Fragment*	• Kemampuan mahasiswa dalam menjelaskan dan mengimlementasikan	BU-1,3 BP-1	5%
6	Mahasiswa	• Layout	• Ceramah	3x50	Dosen menjelaskan	• Kemampuan	BU-1,2	5%

DUMMY

	mampu memmbandingkan penggunaan linear layout, relative layout, constraint layout dan mengimplementasik an penggunaanya pada aplikasi android	manager • Linear layout • Relative layout • Constraint layout	• Tanya Jawab • Latihan/Projec t-based learning • Laporan	menit	tentang layout dan kemudian memberikan contoh program yang memerlukan penggunaan layout. Mahasiswa menganalisis materi layout dan mengimplementasikan penggunaannya	mahasiswa dalam membandingkan dan mengimlementasikan	BP-1	
7	Mahasiswa mampu membandingkan penggunaan dialog, toast, snackbar dan actionbar dan menerapkan penggunaannya pada aplikasi android	• Dialog • Toast • Snackbar • Actionbar	• Ceramah • Tanya Jawab • Projectbased learning • Penugasan	3x50 menit	Dosen menjelaskan tentang dialog, toast, snackbar dan actionbar kemudian memberikan memberikan contoh. Mahasiswa membandingkan penggunaan dialog, toast snackbar, actionbar dan mengimplementasikan penggunaannya	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,3 BP-1	5%
8	**Ujian Tengah Semester**							
9	Mahasiswa mampu menjelaskan konsep *Adapter* dan *RecyclerView*, serta mengimplementasikannya di Android	• *Adapter* • *RecyclerView*	• Ceramah • Tanya Jawab • Latihan/Project-based learning • Penugasan	3x50 menit	Dosen menjelaskan tentang *Adapter* dan recylerview dan kemudian memberikan contoh program yang memerlukan penggunaan *Adapter* dan *RecyclerView*. Mahasiswa memberikan pertanyaan, memahami dan mengimplementasikan penggunaan *Adapter* recylerview.	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,3 BP-2	5%
10	Mahasiswa mampu menjelaskan konsep SearchView, NestedScrollView, CardView, Tabhost dan mengimplementasikannya	• Searchview • Cardview • NestedScrollView • Tabhost	• Ceramah • Tanya Jawab • Latihan/Project-based learning • Penugasan	3x50 menit	Dosen menjelaskan tentang View dan kemudian memberikan contoh program yang penggunaan *Fragment*. Mahasiswa memahami dan mengimplementasikan penggunaan View	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,3 BP-2	5%
11	Mahasiswa mampu menjelaskan notifikasi di android dan mengimplementasikan notifikasi	• Notfikasi Android • Notification Manager • Service notification • Notification *Permission*	• Demonstrasi • Project based learning • Laporan	3x50 menit	Dosen menjelaskan tentang notification dan kemudian memberikan contoh program yang penggunaan *Fragment*. Mahasiswa memahami dan mengimplementasikan penggunaannya	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,3 BP-2	5%
12	Mahasiswa mampu menjelaskan penggunaan Sharedpreference dan	• Sharedpreference • OnSharedPreferenceChangeListener	• Demonstrasi • Project based learning • Laporan	3x50 menit	Dosen menjelaskan tentang Sharedpreference dan kemudian memberikan contoh program.	• Kemampuan mahasiswa dalam memahami dan mengimlementasikan	BU-1,3 BP-2	5%

DUMMY

	mengimplementasi kannya pada aplikasi Android				Mahasiswa memahami dan mengimplementasikan penggunaan Sharedpreference			
13	Mahasiswa mampu menganalisis bagaimana sistem bergerak dapat berkomunikasi dengan Server dan mengimplementasikannya	• Web service • Json • HTTP Request	• Demonstrasi • Project based learning • Laporan	3x50 menit	Dosen menjelaskan kosep web service, penggunaan JSON, HTTP Request dan dan kemudian memberikan contoh program. Mahasiswa menganalisis dan mengimplementasikan penggunaan web service, JSON dan HTTP Request pada aplikasi Android	• Kemampuan mahasiswa dalam menganalisisdan mengimlementasikan	BU-1,3 BP-2	5%
14 - 15	Project Akhir	• Pengembangan aplikasi sistem bergerak •	• Case base project • Pembelajaran mandiri •	3x50 menit	Dosen memberikan bimbingan dan menilai kelayakan awal project akhir yang dirancang oleh mahasiswa. Mahasiswa menentukan topik tugas akhir, melakukan perancangan dan implementasi project akhir. Mahasiswa melakukan uji coba aplikasi dan memastikan kelayakan akhir aplikasi selanjutnya dosen melakukan review dan menetapkan kelayakan project akhir untuk bisa diterima dan diberi penilaian.	• Kemampuan mahasiswa dalam menuntaskan project	BU-1,2,3 BP-1,2	15%
16	**Ujian Akhir Semester**							**15%**

DUMMY

KOMPONEN PENILAIAN:

Ujian Tengah Semester : 15 %
Ujian Akhir Semester : 15 %
Tugas : 10 %
Quiz : 10 %
Case Method : 0 %
Pojcect Base Learning : 50%
Total : 100 %

SISTEM PENILAIAN

Table 3. Sistem Penialaian

	Nilai Mutu	Angka Mutu	Sebutan Mutu		Nilai Mutu	Angka Mutu	Sebutan Mutu
85 – 100	A	4.0	Dengan pujian	55 – 59	C	2.0	Cukup
80 – 84	A-	3.6	Sangat baik sekali	50 – 54	C-	1.6	Kurang cukup
75 – 79	B+	3.3	Baik sekali	40 – 49	D	1.0	Kurang
70 – 74	B	3.0	Baik	≤ 39	E	0.0	Gagal
65 – 69	B-	2.6	Cukup Baik		T	-	Tertunda
60 – 64	C+	2.3	Lebih dari cukup				

DUMMY

BAB 2
PENGENALAN PEMROGRAMAN SISTEM BERGERAK BERBASIS ANDROID

Pemrograman sistem bergerak berbasis Android mengacu pada pengembangan aplikasi untuk perangkat mobile, seperti smartphone dan tablet, menggunakan sistem operasi Android. Android adalah platform yang populer dan mendominasi pasar sistem operasi mobile. Android menawarkan berbagai pustaka dan API yang membantu pengembang dalam menciptakan aplikasi yang interaktif, fungsional, dan user-friendly. Dengan SDK (Software Development Kit) Android, pengembang dapat menggunakan alat-alat seperti Android Studio untuk menulis, menguji, dan mendistribusikan aplikasi di berbagai perangkat Android.

A. Android Studio Judul

Android adalah sistem operasi berbasis Linux yang dirancang khusus untuk perangkat mobile, seperti smartphone dan tablet. Dikembangkan awalnya oleh Android Inc. dan kemudian diakuisisi oleh Google pada tahun 2005, Android kini menjadi salah satu sistem operasi mobile terbesar di dunia.

Android mendukung banyak bahasa pemrograman, seperti Java dan Kotlin, yang memungkinkan pengembang memilih bahasa yang sesuai dengan kebutuhan mereka. Platform ini juga memberikan fleksibilitas tinggi melalui lingkungan open-source, sehingga pengembang dapat memodifikasi dan mengkustomisasi aplikasi dengan bebas.

Sementara itu, Android Studio adalah Integrated Development Environment (IDE) resmi untuk mengembangkan aplikasi Android. Dibangun di atas platform IntelliJ IDEA, Android Studio menawarkan berbagai alat dan fitur yang dirancang untuk meningkatkan produktivitas pengembang dalam membangun aplikasi. Dengan dukungan dari Google, IDE ini menyediakan antarmuka yang user-friendly serta alat debugging, emulasi, dan pengujian yang memungkinkan pengembang menciptakan aplikasi berkualitas tinggi dengan lebih efisien. Beberapa fitur unggulan Android Studio meliputi:

1. **Sistem *Build* Berbasis *Gradle***

   Sistem *Build* yang fleksibel memungkinkan pengembang untuk menyesuaikan proses *Build* dan menciptakan varian aplikasi dengan mudah.

DUMMY

2. **Emulator Cepat dan Kaya Fitur**
   Emulator Android memungkinkan pengujian aplikasi di berbagai konfigurasi perangkat, membantu pengembang memverifikasi kinerja aplikasi.
3. **Lingkungan Terpadu**
   Mendukung pengembangan aplikasi untuk berbagai perangkat Android, seperti smartphone, tablet, TV, dan wearable.
4. **Live Edit**
   Fitur ini memungkinkan pembaruan kode real-time pada emulator atau perangkat fisik.
5. **Template Kode dan Integrasi GitHub**
   Mempermudah pembuatan fitur umum aplikasi dan memungkinkan impor kode sampel.
6. **Alat Lint**
   Membantu memeriksa performa, kegunaan, kompatibilitas versi, serta masalah lainnya.
7. **Dukungan C++ dan NDK**
   Memungkinkan pengembangan aplikasi dengan performa tinggi menggunakan bahasa C++.
8. **Dukungan Google Cloud Platform**
   Memudahkan integrasi dengan layanan Google Cloud, seperti Google Cloud Messaging dan App Engine Ketik konten Sub Bab disini

B. Struktur Proyek di Android Studio

Dalam Android Studio, setiap proyek memiliki struktur yang terdiri dari beberapa modul. Modul-modul ini berisi kode sumber dan *resource* yang diperlukan untuk menjalankan aplikasi Android secara lengkap. Android Studio mengorganisasi modul-modul ini ke dalam satu tampilan proyek yang komprehensif dan intuitif, sehingga memudahkan pengembang dalam mengelola file dan komponen yang sering digunakan.

Secara umum, ada tiga jenis modul yang paling umum digunakan dalam proyek Android:

1. **Modul Aplikasi Android**
   Modul utama yang berisi seluruh kode sumber, layout, gambar, dan konfigurasi inti yang diperlukan untuk membangun dan menjalankan aplikasi Android. Modul ini merupakan bagian wajib dalam proyek Android, karena menjadi wadah keseluruhan aplikasi yang akan dikompilasi menjadi APK.

DUMMY

2. **Modul Library**
   Modul ini digunakan untuk menyimpan pustaka atau komponen yang dapat digunakan ulang dalam aplikasi atau proyek lain. Modul Library memudahkan pengembangan fitur atau elemen serupa yang bisa diimpor ke dalam beberapa aplikasi, mencakup elemen UI, logika bisnis, atau utilitas umum.
3. **Modul Google App Engine**
   Modul khusus yang mendukung integrasi dengan Google Cloud Platform, ideal untuk aplikasi yang memerlukan backend. Modul ini memungkinkan penyimpanan data, menjalankan logika server, dan layanan cloud lainnya, mendukung kebutuhan skalabilitas dan konektivitas server-side secara efisien.

Di Android Studio, tampilan proyek disusun berdasarkan modul, sehingga setiap komponen yang ada dalam proyek dapat diakses dengan mudah dari satu tampilan hierarki. Tampilan ini memberikan pengelompokan berdasarkan fungsi dan tipe file, membantu pengembang dalam mengakses kode sumber, *resource* , dan file konfigurasi tanpa harus membuka setiap folder secara manual. Struktur Project Android meliputi:

1. **Manifest**

   Folder Manifest adalah lokasi dari file AndroidManifest.xml, yang menjadi salah satu file paling penting dalam proyek Android. File ini berfungsi sebagai deskripsi dasar aplikasi kepada sistem Android, memberikan informasi penting yang dibutuhkan agar aplikasi dapat berjalan dengan benar. Di dalam AndroidManifest.xml, pengembang mendefinisikan:
   a. Izin Aplikasi (*Permissions*): Setiap izin yang diperlukan aplikasi, seperti akses ke jaringan, penyimpanan, atau lokasi, didefinisikan di sini.
   b. Aktivitas Utama (*Main Activity*): Aktivitas yang akan pertama kali diluncurkan saat aplikasi dibuka oleh pengguna.
   c. Konfigurasi Dasar Lainnya: Informasi seperti tema aplikasi, orientasi layar, versi API minimal yang didukung, dan fitur yang dibutuhkan.
2. **Java**

   Folder Java berisi kode sumber yang ditulis dalam bahasa Java atau Kotlin, yang menjadi logika utama dari aplikasi. Di dalam folder ini, terdapat tiga subfolder yang memiliki fungsi khusus, yaitu:

DUMMY

a. Main
   Subfolder ini menyimpan kode sumber utama dari aplikasi, termasuk kelas-kelas yang berfungsi sebagai logika bisnis dan pengelolaan data. Subfolder ini menjadi bagian utama tempat pengembang menulis fungsionalitas aplikasi.
b. *Test*
   Subfolder Test digunakan untuk menyimpan kode pengujian unit atau uji coba yang dilakukan terhadap fungsi-fungsi aplikasi. Pengujian ini biasanya dilakukan untuk memastikan bahwa metode atau fungsi tertentu berjalan sesuai dengan yang diharapkan.
c. Android Test
   Subfolder Android Test berisi pengujian yang dirancang untuk berjalan pada perangkat Android atau emulator. Pengujian ini memungkinkan pengembang untuk mengecek performa dan kompatibilitas aplikasi pada berbagai perangkat Android.

3. Res (*resource s*)

Folder Res (*resource* s) adalah folder yang berisi seluruh *resource* non-kode yang digunakan dalam aplikasi, seperti layout, gambar, dan string. Folder ini terbagi menjadi beberapa subfolder untuk mengorganisasi berbagai jenis *resource* , yaitu:

a. Drawable
   Subfolder Drawable menyimpan berbagai gambar atau ikon yang digunakan dalam aplikasi. Format yang sering digunakan untuk gambar di sini adalah PNG, JPEG, atau XML.
b. Layout
   Subfolder Layout berisi file layout dalam format XML, yang mendefinisikan struktur dan tampilan antarmuka pengguna (UI) untuk setiap aktivitas atau komponen aplikasi. Di sini, elemen-elemen UI seperti tombol, teks, dan gambar diatur dan dikelompokkan untuk memberikan tampilan visual yang konsisten.
c. Values
   Subfolder Values berisi file seperti strings.xml, colors.xml, dan styles.xml.
   1) strings.xml berfungsi untuk menyimpan semua teks yang digunakan dalam aplikasi, mempermudah proses terjemahan atau perubahan teks secara keseluruhan.
   2) colors.xml menyimpan palet warna yang digunakan dalam desain aplikasi, sementara

DUMMY

3) styles.xml mengelola gaya tampilan, seperti tema dan atribut visual lainnya, untuk menjaga konsistensi desain aplikasi.

Android Studio menyediakan beberapa opsi tampilan untuk mengorganisir struktur proyek. Secara default, Android Studio menampilkan tampilan Android, yang mengelompokkan file-file proyek berdasarkan modul dan tipe file untuk kemudahan navigasi. Namun, untuk melihat struktur asli file di sistem penyimpanan, pengguna dapat memilih tampilan Project dari menu Project View. Tampilan ini menampilkan file proyek seperti yang disimpan dalam sistem penyimpanan, memperlihatkan semua folder dan subfolder dengan urutan aslinya, tanpa penyusunan berdasarkan modul.

C. Sistem *Build Gradle* di Android Studio

Gradle adalah sistem *Build* utama yang digunakan dalam Android Studio untuk mengelola proses *Build* aplikasi. *Gradle* menyediakan kerangka kerja yang fleksibel dan dapat disesuaikan, yang memungkinkan pengembang untuk membangun aplikasi Android dengan cara yang efisien dan terstruktur. Sistem ini bekerja melalui plugin Android *Gradle* , yang menambahkan kemampuan khusus untuk mengelola proses *Build*, konfigurasi, dan struktur proyek Android dengan lebih mudah.

1. Kustomisasi Proses *Build*
   Pengembang dapat menyesuaikan dan memperluas proses *Build* untuk memenuhi kebutuhan aplikasi.
2. Varian *Build*
   Membuat beberapa versi aplikasi (seperti versi gratis dan berbayar) dari satu proyek.
3. Penggunaan Ulang Kode dan *resource*
   Kode dan *resource* dapat digunakan kembali di seluruh set sumber.

Struktur *Gradle* diatur dalam dua jenis file:

1. *Build. Gradle* (Proyek)
   File *Build. Gradle* di tingkat proyek mengatur konfigurasi global untuk seluruh proyek Android. File ini berisi konfigurasi dasar yang akan berlaku untuk semua modul dalam proyek, seperti repository dependensi yang digunakan, versi plugin Android *Gradle* , dan variabel-variabel global. File ini juga sering digunakan untuk mendefinisikan tugas-tugas yang berlaku di seluruh proyek, seperti

DUMMY

tugas untuk membersihkan atau membangun semua modul dalam proyek sekaligus.

2. *Build*. *Gradle* (Modul)

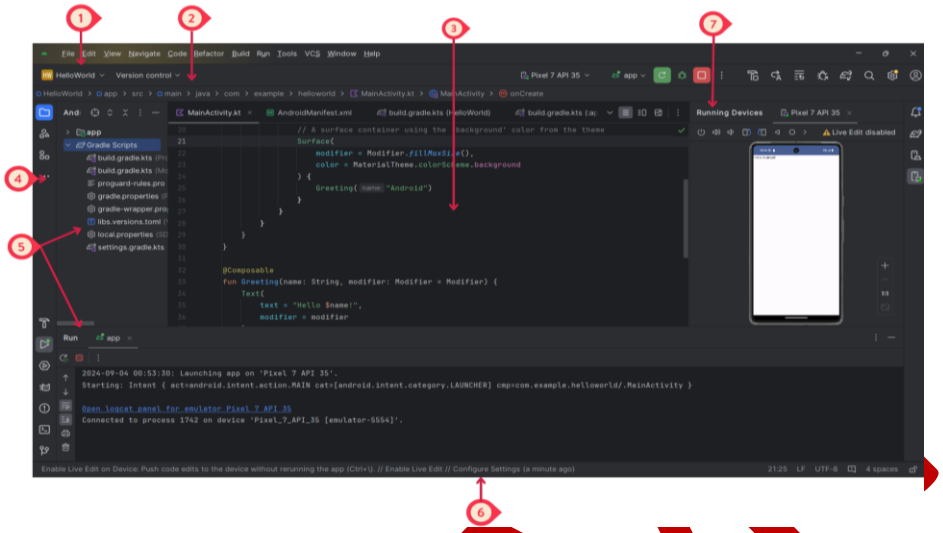
Setiap modul aplikasi atau library memiliki file *Build*. *Gradle* sendiri di tingkat modul. File ini mengatur konfigurasi spesifik untuk modul tersebut, seperti versi SDK minimum dan target, dependensi pustaka yang diperlukan, serta konfigurasi varian *Build*. Pada modul aplikasi, file *Build*. *Gradle* ini juga mengatur *resource* , compile options, dan konfigurasi lain yang terkait dengan modul tersebut. Dalam modul library, file ini lebih terfokus pada konfigurasi yang diperlukan untuk membuat pustaka yang dapat digunakan ulang di dalam proyek lain.

Gradle mendukung pengembangan multi-APK, yang memungkinkan pembuatan APK untuk berbagai konfigurasi perangkat tanpa harus mengubah file sumber utama.

D. Komponen-Komponen di Android Studio

Android Studio adalah lingkungan pengembangan terpadu (Integrated Development Environment/IDE) resmi untuk membangun aplikasi Android. Berdasarkan IntelliJ IDEA, Android Studio menyediakan alat yang lengkap dan intuitif untuk menulis, menguji, dan mempublikasikan aplikasi Android. Untuk memudahkan pengembang dalam bekerja, Android Studio dilengkapi dengan berbagai komponen yang membantu dalam proses pengembangan. Berikut adalah komponen-komponen utama yang ada di Android Studio beserta penjelasannya:

DUMMY



Gambar 1. Komponen Android Studio

Penjelasan Komponen:

1. Toolbar

   Toolbar Android Studio menyediakan akses cepat ke perintah dan aksi umum yang sering digunakan selama proses pengembangan. Beberapa fungsi penting di toolbar termasuk:

   a. Run: Menjalankan aplikasi di perangkat atau emulator.
   b. Debug: Memulai mode debugging untuk aplikasi yang sedang dikerjakan.
   c. AVD Manager: Membuka Android Virtual Device (AVD) Manager, yang memungkinkan pengembang membuat dan mengelola emulator Android.
   d. SDK Manager: Mengatur Android Software Development Kit (SDK), yang mencakup versi Android dan alat-alat pengembangan lainnya.

   Toolbar dirancang untuk mempercepat akses ke fungsi-fungsi penting dalam pengembangan, mengurangi waktu yang diperlukan untuk menemukan dan menjalankan perintah dari menu.

2. Navigasi

   Navigasi di Android Studio adalah mekanisme yang memungkinkan aplikasi berpindah antar layar, aktivitas, atau *Fragment* secara terstruktur. Dalam konteks pengembangan aplikasi, navigasi membantu pengembang untuk mengelola alur pengguna (user flow) dengan lebih mudah dan memastikan pengguna dapat bergerak dari

DUMMY

satu layar ke layar lain sesuai dengan fungsionalitas aplikasi. Dengan menggunakan Navigation Component, pengembang dapat mengatur dan mengelola proses perpindahan ini secara visual dan lebih efisien.

3. Editor Window

   Editor Window adalah area utama tempat pengembang menulis dan mengedit kode. Di dalam jendela ini, pengembang dapat membuka beberapa file secara bersamaan dan beralih di antara file-file tersebut menggunakan tab. Editor Window dilengkapi dengan fitur-fitur yang mempermudah proses penulisan kode, seperti:

   a. Syntax Highlighting: Menandai komponen kode berdasarkan jenisnya (seperti variabel, fungsi, dan komentar), membuat kode lebih mudah dibaca.
   b. Code Completion: Menyediakan saran kode secara otomatis, membantu mengurangi kesalahan pengetikan dan mempercepat penulisan kode.
   c. Refactoring Tools: Memungkinkan pengembang untuk mengubah struktur kode tanpa mengubah fungsionalitas, seperti mengganti nama variabel atau memindahkan metode ke kelas lain.

4. Tool Window (Panel Jendela Fitur)

   Panel Jendela Fitur, atau Tool Windows, adalah area di Android Studio yang memberikan akses cepat ke berbagai alat dan fitur yang mendukung pengembangan aplikasi. Di sini, pengembang dapat membuka jendela-jendela seperti Project, Logcat, Android Profiler, Device File Explorer, dan lainnya. Tool Windows memungkinkan pengembang untuk melakukan tugas-tugas seperti melihat struktur proyek, memantau aktivitas aplikasi, dan menganalisis performa aplikasi secara mudah.

5. Jendela Fitur

   Jendela Fitur di Android Studio adalah bagian dari antarmuka pengguna yang menyediakan akses ke berbagai alat dan informasi yang diperlukan dalam pengembangan aplikasi. Jendela fitur ini meliputi komponen-komponen seperti Project Window, yang menampilkan struktur file proyek, dan Logcat, yang menampilkan log dari aplikasi yang sedang berjalan. Jendela fitur memudahkan pengembang dalam mengakses informasi penting yang dibutuhkan selama proses pengembangan.

6. Status Bar

   Status Bar di Android Studio menampilkan informasi penting terkait proyek dan proses yang sedang berjalan. Status bar terletak di

DUMMY

bagian bawah layar dan menunjukkan berbagai detail, seperti status sinkronisasi *Gradle* , progres *Build*, atau peringatan proyek. Status Bar memungkinkan pengembang untuk memantau status proyek secara real-time, membantu dalam mendeteksi masalah atau melihat status terkini dari proses *Build* dan sinkronisasi.

7. Running Device

   Fungsi Running Device di Android Studio adalah untuk mengeksekusi dan menjalankan aplikasi yang sedang dikembangkan pada perangkat yang dipilih, baik itu perangkat fisik yang terhubung melalui USB atau emulator yang disediakan oleh Android Studio. Running Device memungkinkan pengembang untuk melihat hasil aplikasi secara langsung di perangkat dan melakukan pengujian fungsionalitas aplikasi secara real-time, yang sangat penting untuk memastikan aplikasi berjalan dengan benar sebelum dipublikasikan.

E. Android Virtual Device (AVD)

Android Virtual Device (AVD) adalah emulator yang disediakan oleh Android Studio untuk menjalankan dan menguji aplikasi tanpa memerlukan perangkat fisik. AVD memungkinkan pengembang untuk meniru berbagai konfigurasi perangkat Android, termasuk ukuran layar, versi OS, dan perangkat keras lainnya seperti kamera dan sensor. Langkah Membuat AVD:

1. Buka Android Studio dan pilih “AVD Manager” dari menu “Tools”.
2. Pilih “Create Virtual Device” dan pilih tipe perangkat yang ingin digunakan, misalnya smartphone, tablet, atau wearable.
3. Pilih versi Android yang akan digunakan (biasanya disesuaikan dengan target API aplikasi).
4. Konfigurasi spesifikasi perangkat seperti ukuran RAM dan resolusi layar.
5. Klik “Finish” untuk menyelesaikan konfigurasi.
6. AVD sekarang dapat dijalankan untuk menguji aplikasi. Emulator ini menampilkan layar perangkat Android yang menjalankan aplikasi seolah-olah di perangkat fisik.

F. Android SDK (Software Development Kit)

Android SDK (Software Development Kit) adalah kumpulan alat yang disediakan oleh Google untuk membantu pengembang dalam

DUMMY

membangun aplikasi yang kompatibel dengan sistem operasi Android. SDK Android menjadi fondasi bagi setiap proyek aplikasi Android, menyediakan pustaka, API, dan alat tambahan yang memungkinkan pengembang untuk menciptakan aplikasi yang kaya fitur dan dapat berinteraksi dengan berbagai komponen perangkat. SDK ini menjadi bagian esensial dari ekosistem Android, mendukung seluruh proses pengembangan mulai dari pembuatan antarmuka pengguna hingga pengujian aplikasi pada perangkat.

Berikut adalah beberapa komponen utama yang terdapat dalam Android SDK:

1. API untuk Komponen UI: Seperti tombol, menu, dan teks untuk membangun antarmuka pengguna (UI).
2. Library Perangkat Keras: Untuk mengakses fitur perangkat keras seperti kamera, mikrofon, dan sensor.
3. Dokumentasi dan Kode Contoh: SDK Android dilengkapi dengan dokumentasi yang luas serta contoh kode untuk membantu pengembang memahami implementasi fungsi-fungsi dasar.
4. Android Debug Bridge (ADB): Alat untuk melakukan debugging dan pengujian aplikasi pada perangkat fisik maupun emulator.

G. Android Runtime (ART)

Android Runtime (ART), adalah lingkungan eksekusi aplikasi di Android yang menggantikan Dalvik Virtual Machine pada Android versi 5.0 (Lollipop) dan versi selanjutnya. ART bertanggung jawab untuk menjalankan aplikasi Android dengan beberapa keunggulan kinerja:

1. Kompilasi Ahead-of-Time (AOT)
   ART menggunakan kompilasi AOT untuk mengurangi waktu startup aplikasi, yang meningkatkan performa saat aplikasi dijalankan.
2. Format DEX (Dalvik Executable)
   ART mengeksekusi bytecode dalam format DEX, yang dioptimalkan untuk sistem Android.
3. Garbage Collection (GC)
   ART memiliki sistem garbage collection yang lebih efisien, yang membantu aplikasi berjalan dengan lebih cepat dan mengurangi penggunaan memori yang tidak diperlukan.

H. Alur Hidup Aplikasi Android (Lifecycle)

DUMMY

Alur hidup aplikasi Android dikendalikan oleh sistem Android, yang menentukan kapan aplikasi dimuat atau dihentikan berdasarkan kondisi sistem dan aktivitas pengguna. Proses ini melibatkan beberapa tingkat prioritas:

1. Proses Latar Depan: Proses dengan prioritas tertinggi yang aktif berinteraksi dengan pengguna, seperti aplikasi yang sedang terbuka.
2. Proses yang Terlihat: Proses yang terlihat di layar tetapi tidak aktif, misalnya aplikasi yang diminimalkan.
3. Proses Layanan: Proses yang menjalankan layanan di latar belakang, seperti pemutaran musik.
4. Proses dalam Cache: Proses yang disimpan dalam cache dan siap digunakan kembali jika dibutuhkan, meski tidak terlihat oleh pengguna.

Prioritas ini memungkinkan Android untuk mengelola memori secara efisien dan mempertahankan pengalaman pengguna yang responsif meskipun terdapat banyak aplikasi yang berjalan secara bersamaan.

DUMMY

I. Prerequisite dan Instalasi

1. Prerequisite

Prerequisite adalah prasyarat yang diperlukan untuk bisa menjalankan aplikasi. Dalam hal ini untuk bisa membuat dan menjalankan android studio menggunakan bahasa pemrograman Kotlin maka dibutuhkan persyaratan perangkat keras dan perangkat lunak. Berikut adalah rincian kebutuhan perangkat keras dan lunak yang dibutuhkan:

Table 4. Spesifikasi Perangkat Keras

Nama	Minimum requirement yang disarankan
Memory	8 GB
Hard Disk	500 GB
Processor	Intel core i3 gen 13 atau amd ryzen 5
VGA	Internal dengan dukungan OpenGL ES

Table 5. Spesifikasi Perangkat Lunak

Nama	Spesifikasi
Sistem Operasi	Windows 10, macOS10.14 atau linux 64 bit
JDK	Versi 11
SDK	Versi 21 atau lebih tinggi (Buku ini menggunakan SDK 34)
Android Emulator	-
Dependencies lainnya	-

DUMMY

2. Instalasi

Untuk instalasi silahkan lanjutkan dengan melakukan langkah berikut:

a. Unduh Android Studio dari situs resmi Android Developer. Pilih versi terbaru, setujui syarat dan ketentuan, lalu unduh file instalasinya.
b. Buka file installer yang telah diunduh (biasanya di folder Downloads) dengan mengklik dua kali untuk memulai proses instalasi.
c. Ikuti petunjuk di Android Studio Setup Wizard. Pilih konfigurasi Standard yang direkomendasikan, lalu pilih tema tampilan (Dark atau Light) dan klik Finish untuk melanjutkan.
d. Setelah itu, Android Studio akan mengunduh Android SDK dan komponen lain yang diperlukan. Tunggu hingga proses ini selesai.
e. Setelah semua komponen terinstal, Android Studio akan terbuka di halaman Welcome to Android Studio. Pilih New Project untuk membuat proyek baru dan memastikan instalasi sudah benar.
f. (Opsional) Tambahkan sdk\platform-tools ke variabel PATH agar Anda bisa menjalankan perintah Android SDK dari command prompt. Buka Control Panel > System and Security > System > Advanced system settings, pilih Environment Variables, lalu tambahkan lokasi direktori platform-tools ke variabel PATH, misalnya: C:\Users\Username\AppData\Local\Android\Sdk \platform-tools.

g. Android Studio kini siap digunakan untuk mengembangkan aplikasi Android di Windows.

J. Tugas

1. Eksplorasi: Buat screen layar tampilan komponen android studio yang telah anda install dan uraikan komponen sesuai dengan fungsinya .
2. Laporan: Buat laporan dengan melampirkan screenshot hasil instalasi android studio dan berikan penjelasan untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 3
ACTIVITY DAN *INTENT*

Dalam dunia pengembangan aplikasi Android, pemahaman tentang komponen dasar seperti *Activity*, *Permission*, *Feature*, dan Intent sangat penting. Komponen-komponen ini tidak hanya membentuk fondasi dari setiap aplikasi, tetapi juga menentukan interaksi pengguna dengan aplikasi tersebut. Dengan memahami konsep-konsep ini, para pengembang dapat merancang aplikasi yang lebih efisien dan responsif terhadap kebutuhan pengguna. Bab ini akan membahas tentang masing-masing konsep, dimulai dari *Activity* sebagai elemen utama antarmuka pengguna, diikuti oleh *Permission* yang mengatur akses aplikasi terhadap sumber daya perangkat, *Feature* yang mendefinisikan kemampuan aplikasi, serta Intent yang memungkinkan komunikasi antar komponen.

A. *Activity*

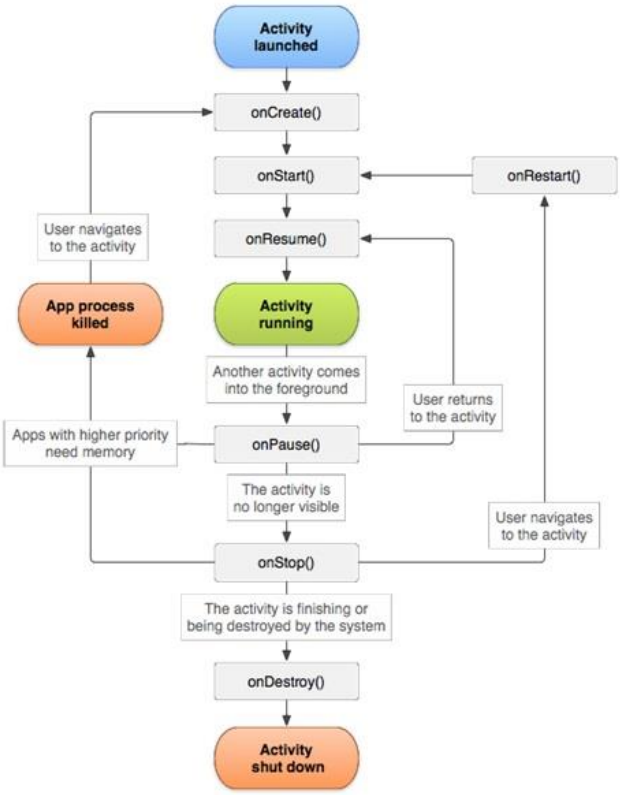
Activity adalah komponen yang dapat dilihat oleh pengguna, memungkinkan mereka untuk berinteraksi dengan aplikasi. Jika dibandingkan dengan aplikasi web dan desktop, *Activity* memiliki kesamaan dengan halaman dan form. Dalam konteks ini, *Activity* berfungsi sebagai antarmuka pengguna yang memfasilitasi interaksi tersebut. Namun, *Activity* tidak bersifat permanen; ia dapat diciptakan dan dihancurkan sesuai kebutuhan aplikasi.

Setiap *Activity* memiliki siklus hidup (lifecycle) yang menggambarkan kondisi yang dialami selama proses dari diciptakan hingga dihancurkan. *Activity* mewakili satu layar dalam aplikasi Android, dan setiap *Activity* dapat dianggap sebagai tampilan yang dirancang untuk interaksi pengguna. Misalnya, ketika pengguna membuka aplikasi, mereka biasanya akan melihat *Activity* pertama yang ditampilkan, yang bisa berupa halaman beranda atau menu utama. *Activity* memiliki siklus hidup yang unik, di mana status dan perilakunya berubah tergantung pada interaksi pengguna dan tindakan aplikasi itu sendiri.

Sebagai komponen utama yang ada di Android Studio, *Activity* berfungsi untuk menampilkan antarmuka pengguna (UI) dari aplikasi, yang biasanya diatur dengan metode setContentView(). Selain menampilkan UI, *Activity* juga digunakan untuk melakukan berbagai kegiatan yang diperlukan di dalam aplikasi, seperti berpindah dari satu tampilan ke tampilan lainnya, menjalankan program lain, dan banyak fungsi lainnya yang dapat dilakukan dalam sebuah *Activity*.

DUMMY

Activity memiliki siklus hidup, atau yang biasanya disebut "lifecycle", yang menggambarkan bagaimana *Activity* bereaksi terhadap peristiwa tertentu, seperti saat dibuka atau ditutup. Siklus hidup ini terdiri dari beberapa metode penting yang diwakili oleh callback. Hal ini dapat dilihat pada Gambar di bawah ini:Ketik konten Sub Bab disini



Gambar 2. Activity Lifecycle
Sumber: developer.android.com

Siklus hidup *Activity* adalah serangkaian metode yang dipanggil oleh sistem untuk mengelola aktivitas. Metode ini memberikan informasi kepada *Activity* tentang keadaan saat ini dan kapan *Activity* perlu melakukan tindakan tertentu. Berikut adalah metode penting dalam siklus hidup *Activity*:

1. **onCreate**(): Metode ini dipanggil saat *Activity* pertama kali dibuat. Pada tahap ini, biasanya kita melakukan inisialisasi UI dan pengaturan awal.

2. **onStart**(): Dipanggil ketika *Activity* mulai terlihat oleh pengguna. Di sini, kita dapat memulai proses yang diperlukan untuk menampilkan konten.
3. **onResume**(): Dipanggil ketika *Activity* mulai berinteraksi dengan pengguna. Ini adalah titik di mana *Activity* berada di latar depan, dan pengguna dapat berinteraksi secara penuh.
4. **onPause**(): Dipanggil ketika *Activity* tidak lagi berada di depan. Pada tahap ini, kita harus menyimpan data penting dan menghentikan aktivitas yang tidak perlu, seperti animasi atau pemutaran video.
5. **onStop**(): Dipanggil ketika *Activity* tidak lagi terlihat oleh pengguna. Di sini, kita dapat membersihkan sumber daya yang tidak perlu dan menyimpan data jika diperlukan.
6. **onDestroy**(): Dipanggil sebelum *Activity* dihancurkan. Ini adalah kesempatan terakhir untuk membebaskan sumber daya dan menyimpan data sebelum *Activity* ditutup sepenuhnya.

Siklus hidup *Activity* menunjukkan bagaimana *Activity* bereaksi terhadap berbagai peristiwa, seperti saat dibuka, berada di latar depan, atau ditutup. Selain itu, berikut adalah beberapa status yang dilalui *Activity* selama siklus hidupnya:

1. **Foreground *Activity***: Tampilan yang paling terlihat oleh pengguna saat menggunakan aplikasi.
2. **Visible *Activity***: Tampilan yang terlihat oleh pengguna tetapi bukan yang paling depan, seperti saat dialog muncul.
3. **Background *Activity***: Tampilan yang disimpan untuk ditampilkan kembali ketika pengguna melakukan interaksi yang sesuai, seperti saat login ke aplikasi.
4. **Empty Process**: Proses yang tidak tampak, namun berfungsi untuk mengubah tampilan aplikasi, seperti Service dan BroadcastReceiver.

B. *Permission*

Permission dalam konteks aplikasi Android adalah izin yang harus diminta oleh aplikasi untuk mengakses sumber daya tertentu di perangkat, seperti data sensitif pengguna, kamera, lokasi, dan penyimpanan. Hal ini penting untuk menjaga privasi pengguna dan melindungi data mereka. *Permission* adalah bagian krusial dari keamanan Android, memastikan bahwa aplikasi hanya dapat mengakses informasi dan sumber daya yang

DUMMY

diperlukan untuk fungsinya, dan memberikan kontrol kepada pengguna atas data pribadi mereka.

Jenis-Jenis *Permission*

Permission di Android dibagi menjadi dua kategori utama:

1. Normal *Permission*s, izin ini dianggap tidak berbahaya bagi pengguna dan dapat diberikan secara otomatis tanpa memerlukan persetujuan eksplisit. Contohnya termasuk akses ke internet dan akses ke penyimpanan yang tidak sensitif.
2. Dangerous *Permission*s, izin ini berpotensi berbahaya bagi privasi pengguna atau akses data, sehingga memerlukan persetujuan eksplisit dari pengguna sebelum diberikan. Contohnya termasuk akses ke lokasi, kontak, dan kamera. Pengembang perlu meminta izin ini secara dinamis saat aplikasi dijalankan.

Permission harus dideklarasikan dalam file AndroidManifest.xml agar aplikasi dapat mengakses sumber daya tertentu. Berikut adalah contoh deklarasi *Permission* untuk akses internet dan lokasi:

```
<uses-Permission
android:name="android.Permission.INTERNET"/>

<uses-Permission
android:name="android.Permission.ACCESS_FINE_LOCATION"/>
```

C. *Features*

Features adalah fungsi atau kemampuan yang ditawarkan oleh aplikasi. Agar aplikasi memberikan pengalaman terbaik bagi pengguna, penting bagi pengembang untuk menentukan *Features* apa saja yang dibutuhkan dan digunakan oleh aplikasi. Sebagai contoh, jika aplikasi memerlukan akses ke kamera, maka *feature* tersebut harus dinyatakan secara eksplisit dalam file AndroidManifest.xml.

Pengembang dapat mengidentifikasi *Features* yang dibutuhkan aplikasi dengan menambahkan tag <uses-*feature*> di dalam file AndroidManifest.xml. Contoh berikut menunjukkan bagaimana mendeklarasikan bahwa aplikasi memerlukan kamera dan GPS:

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

DUMMY

```
<uses-feature
android:name="android.hardware.location.gps"android:require
d="false"/>
```

Android memungkinkan pengembang mendeklarasikan kebutuhan perangkat keras atau perangkat lunak yang spesifik untuk menjalankan aplikasi. Deklarasi ini membantu menyaring perangkat yang kompatibel untuk mengunduh aplikasi, sehingga menjaga pengalaman pengguna tetap optimal. Berikut ini manfaat deklarasi *Features* dalam AndroidManifest.xml:

1. *Features* Perangkat Keras dan Perangkat Lunak: Pengembang bisa menambahkan *Features* tertentu yang dibutuhkan, seperti kamera, GPS, atau sensor lain, langsung dalam file manifest.
2. Kompatibilitas Perangkat: Jika perangkat tidak memiliki *feature* yang dibutuhkan, *Google Play* tidak akan menampilkan atau mengizinkan pengunduhan aplikasi tersebut. Dengan demikian, hanya perangkat yang mendukung semua *Features* penting aplikasi yang bisa menginstalnya, menjaga agar pengguna hanya mendapatkan aplikasi yang bekerja dengan optimal di perangkat mereka.

D. *Intent*

Intent adalah sebuah objek yang mendeskripsikan operasi yang akan dilakukan dalam sistem Android. Dalam konteks pemrograman Android, Intent berfungsi sebagai mekanisme untuk berkomunikasi antara berbagai komponen aplikasi, termasuk antar aktivitas, layanan, dan penerima broadcast. Secara teknis, Intent adalah kelas dalam paket android.content yang memungkinkan aplikasi untuk memulai komponen lain dalam sistem Android, baik itu di dalam aplikasi yang sama atau di aplikasi lain. Dengan menggunakan metode start *Activity*(), Anda dapat memulai aktivitas baru dan mengalihkan kontrol ke aktivitas tersebut. Intent dibagi menjadi dua kategori utama:

1. Explicit Intent

Explicit Intent digunakan untuk memulai komponen tertentu dalam aplikasi yang sama. Biasanya, Explicit Intent digunakan untuk beralih antara aktivitas dalam aplikasi atau untuk memulai layanan dengan jelas menunjuk ke kelas tujuan. Explicit Intent secara langsung menunjuk ke komponen tertentu yang ingin dimulai, biasanya

DUMMY

menggunakan nama kelas dari komponen tersebut. Misalnya, jika ingin memulai aktivitas baru di dalam aplikasi yang sama, akan menggunakan Explicit Intent. Ini sangat berguna ketika kita tahu dengan pasti komponen yang ingin dipanggil. Berikut adalah contoh penggunaan Explicit Intent untuk memulai aktivitas baru:

```
val intent = Intent(this, Target Activity::class.java)
intent.putExtra("EXTRA_MESSAGE", "Hello World") start
Activity(intent)
```

Dalam contoh di atas, Intent dibuat untuk memulai Target *Activity* dan juga mengirimkan data tambahan menggunakan metode putExtra().

2. **Implicit Intent**

Implicit Intent digunakan untuk melakukan tindakan tanpa menunjuk secara langsung ke komponen tertentu. Sebaliknya, Implicit Intent menyatakan tindakan yang diinginkan dan memungkinkan sistem Android untuk memilih aplikasi atau komponen yang sesuai untuk menangani permintaan tersebut. Ini sering digunakan untuk berinteraksi dengan aplikasi lain di perangkat, seperti membuka halaman web di browser atau menampilkan peta. Berikut adalah contoh penggunaan Implicit Intent untuk membuka halaman web:

```
val intent = Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.example.com")) start
Activity(intent)
```

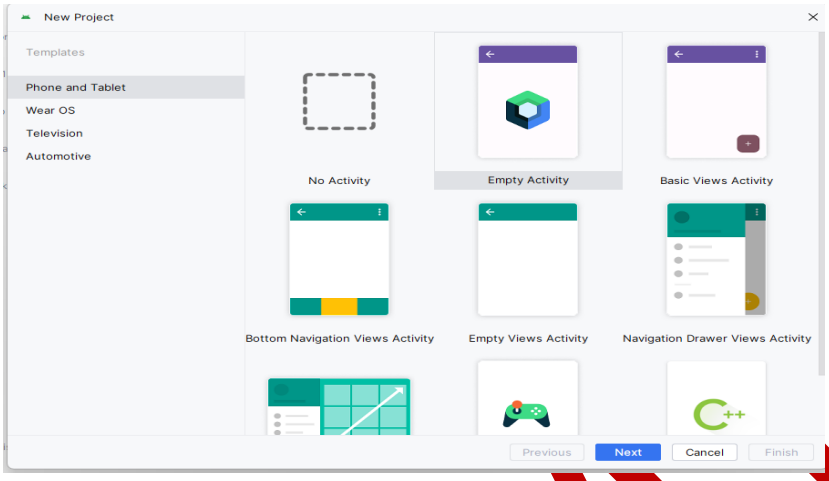
Dalam contoh ini, Intent dibuat dengan tindakan ACTION_VIEW dan URL yang ingin ditampilkan. Sistem Android akan memilih aplikasi yang dapat menangani permintaan ini, seperti browser web.Ketik konten Sub Bab disini

E. Implementasi *Activity* dan *Intent*

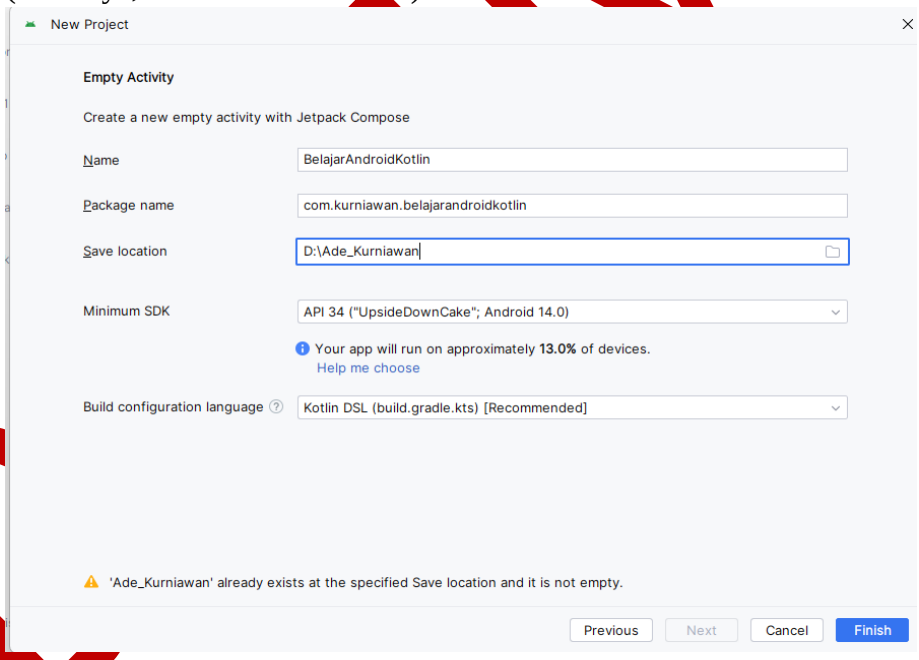
Langkah - Langkah Implementasi Aplikasi:

1. **Membuat Proyek Android Baru**
   Langkah 1.1 : Buka Android Studio dan pilih **Start a new Android Studio project**
   Langkah 1.2: Pilih template Empty *Activity*.

DUMMY

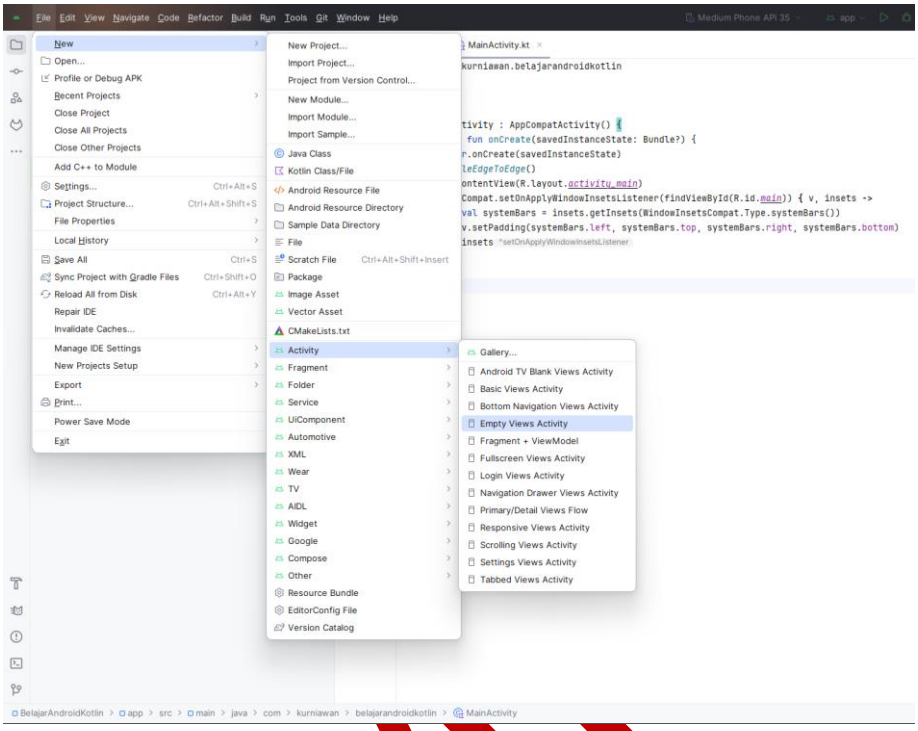


Langkah 1.3: Isi nama proyek, misalnya **BelajarKotlin**, dan pilih *Build* configuration language Kotlin DSL. Pilih Minimum API Level (misalnya, API 24: Android 7.0)

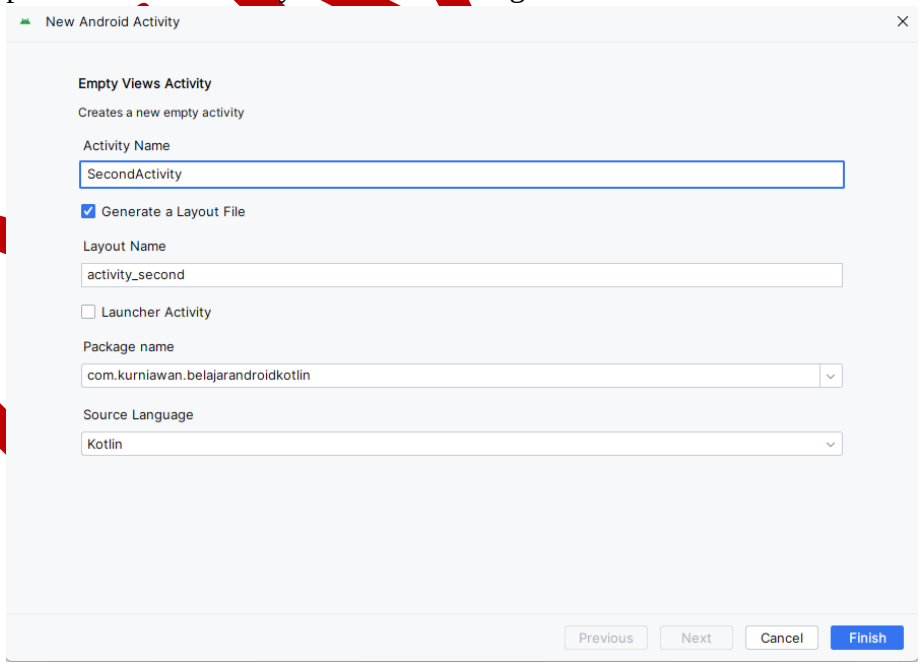


2. **Membuat *Activity* Baru**

   Langkah 2.1: Klik kanan pada package misalnya **com.kurniawan.belajarkotlin, pilih New > *Activity* > Empty *Activity*.**



Langkah 2.2: Beri nama *Activity* baru misalnya **Second *Activity***, dan pastikan Generate Layout File dicentang.



Langkah 2.3: Klik Finish untuk membuat *Activity* baru.

3. **Menambahkan Komponen UI pada Layout**

Langkah 3.1: Tambahkan Button di layout ***Activity*_main.xml** sehingga kodenya menjadi seperti di bawah berikut:

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <Button
        android:id="@+id/button_start"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second Activity"
        android:layout_centerInParent="true"/>
</LinearLayout>
```

Langkah 3.2: Buka kembali ***Activity*_second.xml** dan lakukan perubahan script menjadi seperti di bawah ini:

activity_second.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">
  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Ini Adalah Halaman Second Activity">
  </TextView>
</LinearLayout>
```

4. **Menghubungkan Komponen UI dengan *Activity***

Langkah 4.1: Buka file ***Main Activity.kt***

Langkah 4.2: Tambahkan kode line 20 s.d 25 berikut di dalam method onCreate() untuk menghubungkan tombol dengan fungsionalitas berpindah *Activity*:

activity_main.xml | SecondActivity.kt | MainActivity.kt

```
1   package com.kurniawan.belajarandroidkotlin
2
3   import ...
7
8   class MainActivity : AppCompatActivity() {
9
10      override fun onCreate(savedInstanceState: Bundle?) {
11          super.onCreate(savedInstanceState)
12          setContentView(R.layout.activity_main)
13
14          val buttonStart = findViewById<Button>(R.id.button_start)
15          buttonStart.setOnClickListener { it: View!
16              val intent = Intent( packageContext: this, SecondActivity::class.java)
17              startActivity(intent)
18          }
19      }
20  }
```

Langkah 4.3: Tambahkan kode theme pada ***Activity* Second *Activity* di AndroidManifest.xml** sehingga kodenya menjadi seperti berikut.

DUMMY

activity_main.xml | SecondActivity.kt | MainActivity.kt | AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" >

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="BelajarAndroidKotlin"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        tools:targetApi="31" >

        <!-- SecondActivity -->
        <activity
            android:name=".SecondActivity"
            android:exported="false" />

        <!-- MainActivity -->
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="BelajarAndroidKotlin"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

5. Membuat Project Explicit Intent dan Implicit Intent

Langkah 5.1: Perbarui kode pada **Main *Activity*.kt**

activity_second.xml MainActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val buttonStart = findViewById<Button>(R.id.button_start)
        buttonStart.setOnClickListener {
            val intent = Intent( packageContext: this, SecondActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan3 = findViewById<Button>(R.id.button_pertemuan3)
        buttonPertemuan3.setOnClickListener {
            val intent = Intent( packageContext: this, IntentActivity::class.java)
            startActivity(intent)
        }
```

Langkah 5.2: Buat *Activity* baru dengan nama **Intent *Activity*.kt** dan buatlah kode seperti di bawah berikut :

IntentActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity

class IntentActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge() // Mengaktifkan tampilan edge-to-edge
        setContentView(R.layout.activity_intent_activity) // Mengatur konten layout

        // Mencari Button dengan ID dan menetapkan OnClickListener untuk Explicit Activity
        val buttonExplicit = findViewById<Button>(R.id.button_intent_explicit)
        buttonExplicit.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, ExplicitActivity::class.java)
            startActivity(intent) // Memulai ExplicitActivity
        }

        // Mencari Button dengan ID dan menetapkan OnClickListener untuk Implicit Activity
        val buttonImplicit = findViewById<Button>(R.id.button_intent_implicit)
        buttonImplicit.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, ImplicitActivity::class.java)
            startActivity(intent) // Memulai ImplicitActivity
        }
    }
}
```

Langkah 5.3: Buat *Activity* baru dengan nama **Explicit *Activity*.kt**

IntentActivity.kt | ExplicitActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R
import com.kurniawan.belajarkotlin.ResultExplicitActivity

class ExplicitActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_explicit)

        // Mencari Button dengan ID yang benar
        val buttonSendMessage = findViewById<Button>(R.id.send_message_via_intent)

        // Menetapkan OnClickListener pada Button
        buttonSendMessage?.setOnClickListener { it: View!
            // Membuat Intent untuk berpindah ke ResultExplicitActivity
            val intent = Intent( packageContext: this, ResultExplicitActivity::class.java)
            intent.putExtra( name: "EXTRA_MESSAGE",  value: "Ini adalah message dari Explicit Activity")
            startActivity(intent)

        }
    }
}
```

Langkah 5.4: Buat *Activity* baru dengan nama **Implicit *Activity*.kt**

tivity.kt | activity_main.xml | User.kt | activity_menu_view.xml | activity_tab_host.xml | activity_implicit.xml | ResultExplicit

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class ImplicitActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_implicit) // Pastikan layout sudah benar

        // Tombol untuk membuka aplikasi telepon dan melakukan panggilan
        val buttonAccessContact = findViewById<Button>(R.id.access_contact)
        buttonAccessContact.setOnClickListener {
            val intent = Intent(Intent.ACTION_DIAL) // Gunakan ACTION_DIAL untuk membuka dialer tanpa langsung memanggil
            intent.data = Uri.parse( uriString: "*888#")
            startActivity(intent)
        }
    }
}
```

Langkah 5.5: Buat *Activity* baru dengan nama **ResultExplicit *Activity*.kt**

IntentActivity.kt | ExplicitActivity.kt | ImplicitActivity.kt | ResultExplicitActivity.kt

```kotlin
package com.kurniawan.belajarkotlin

import ...

class ResultExplicitActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_result_explicit)

        // Menerima pesan dari Intent
        val message = intent.getStringExtra( name: "EXTRA_MESSAGE")

        // Menampilkan pesan di TextView
        val textView = findViewById<TextView>(R.id.textViewMessage)
        textView.text = message
    }
}
```

6. **Menambahkan Komponen UI pada Layout**

Langkah 6.1: Isikan Layout dengan ***Activity*_main dan** Root Tag LinearLayout dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:layout_marginTop="24dp"
 android:padding="16dp">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Belajar Basic Android dengan Kotlin"
 android:layout_marginTop="10dp"
 android:layout_marginLeft="10dp"
 android:textSize="15sp"
 android:textStyle="bold"
 android:gravity="center"/>

 <!-- Linear Layout untuk Button Start -->
 <LinearLayout
 android:layout_width="match_parent"
```

DUMMY

```
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:gravity="center">

<Button
 android:id="@+id/button_start"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Second Activity" />
</LinearLayout>

<!-- Linear Layout untuk Button Pertemuan 3 -->
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:gravity="center">

<Button
 android:id="@+id/button_pertemuan3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Intent Activity" />
</LinearLayout>
```

Langkah 6.2: Isikan Layout dengan ***Activity*_intent dan** Root Tag LinearLayout dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="10dp">

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="INTENT - Intent Explicit dan Intent
Implicit"
 android:layout_marginTop="50dp"
 android:layout_marginStart="10dp"
 android:textSize="15sp"
 android:textStyle="bold"
 android:gravity="center" />
```

DUMMY

```
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:gravity="center">

<Button
 android:id="@+id/button_intent_explicit"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Intent Explicit" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:gravity="center">

<Button
 android:id="@+id/button_intent_implicit"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Intent Implicit" />
</LinearLayout>

</LinearLayout>
```

Langkah 6.3: Isikan Layout dengan ***Activity_explicit*** dan Root Tag LinearLayout dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="16dp">  <!-- Menambahkan padding di
tepi -->

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="INTENT - Explicit Activity"
 android:layout_marginTop="50dp"
 android:gravity="center"
 android:textSize="18sp"
 android:textStyle="bold" />
```

RAHMY

```
  <Button
  android:id="@+id/send_message_via_intent"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Kirim Pesan"
  android:layout_gravity="center"
  android:layout_marginTop="20dp"/>
 </LinearLayout>
```

Langkah 6.4: Isikan Layout dengan ***Activity_implicit*** dan Root Tag LinearLayout dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
 xmlns:android="http://schemas.android.com/apk/res/androi
 d"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:padding="16dp"
  android:gravity="center"> <!-- Posisikan di tengah -->

  <TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="INTENT - Intent Implicit"
  android:layout_marginTop="32dp"
  android:textSize="18sp"
  android:textStyle="bold"
  android:gravity="center"
 android:textColor="@android:color/black"  /> <!-- Teks
 tebal, besar, di tengah -->

  <LinearLayout
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_marginTop="16dp"
  android:gravity="center"> <!-- Posisikan tombol di
 tengah -->

  <Button
   android:id="@+id/access_contact"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Access Contact: 0813 - 7448 - 5205"
   android:padding="12dp"
   android:elevation="6dp"  /> <!-- Elevasi untuk efek
 bayangan -->
  </LinearLayout>
</LinearLayout>
```

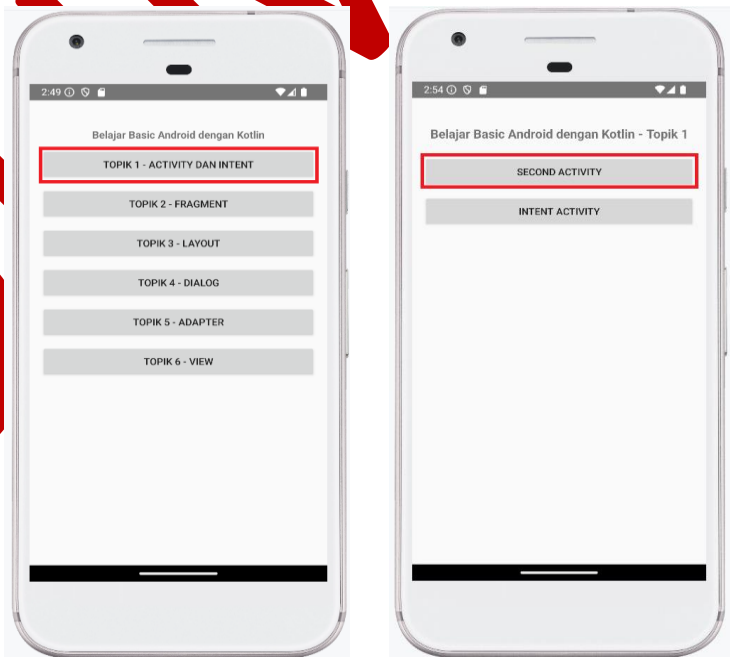
Langkah 6.5: Isikan Layout dengan ***Activity_result_explicit*** dan Root Tag LinearLayout dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:gravity="center"
 android:padding="16dp">

 <TextView
 android:id="@+id/textViewMessage"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Pesan akan muncul di sini"
 android:textSize="18sp" />
</LinearLayout>
```

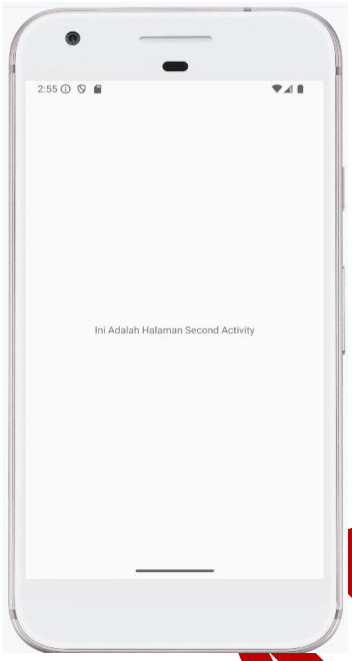
7. Menjalankan dan Menguji Aplikasi

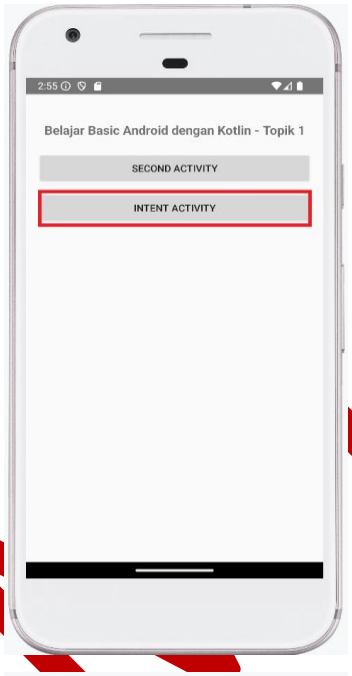
Langkah 7.1: Klik Run atau tekan Shift + F10 untuk menjalankan aplikasi.

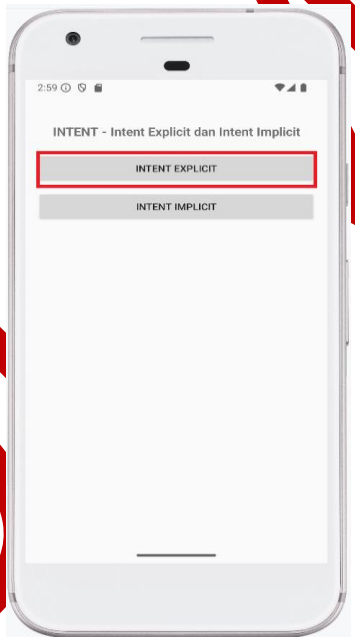
Langkah 7.2: Pilih emulator atau perangkat fisik untuk menjalankan aplikasi.

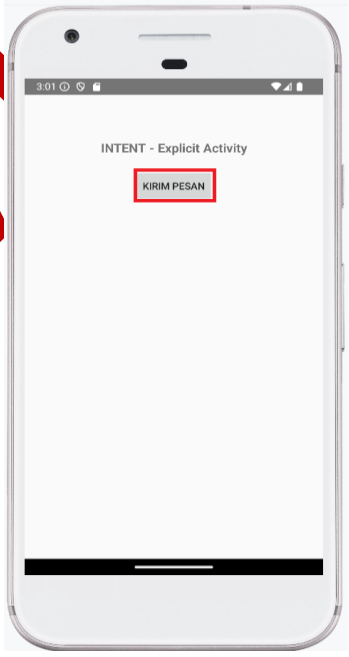


DRAFT

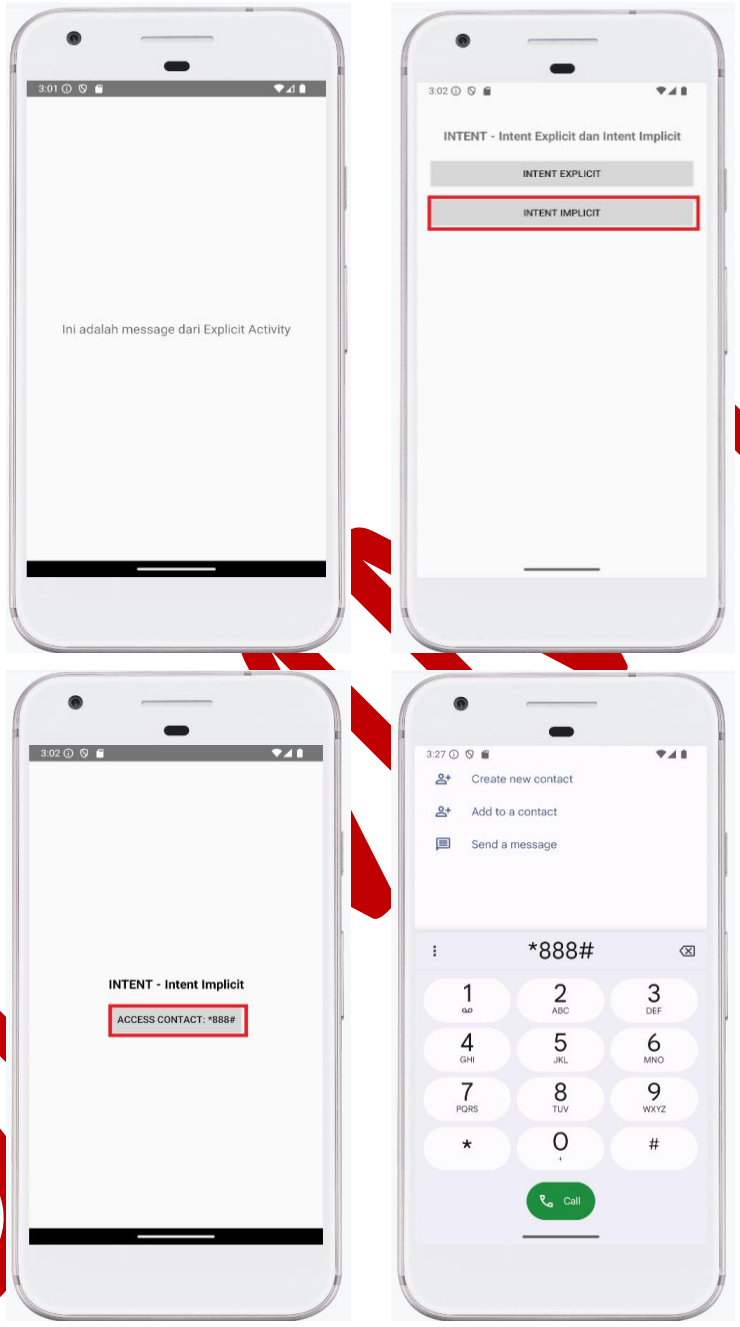

2:55
Ini Adalah Halaman Second Activity


2:55
Belajar Basic Android dengan Kotlin - Topik 1
SECOND ACTIVITY
INTENT ACTIVITY


2:59
INTENT - Intent Explicit dan Intent Implicit
INTENT EXPLICIT
INTENT IMPLICIT


3:01
INTENT - Explicit Activity
KIRIM PESAN

Draft



Gambar 3. Run Aplikasi *Activity* dan *Intent*

F. Tugas

1. Modifikasi: Tambahkan satu *Activity* lagi, misalnya Third *Activity*, dan buat tombol pada Second *Activity* untuk berpindah ke Third *Activity*.
2. Eksplorasi: Ubah teks pada tombol sesuai dengan aktivitas yang dituju.
3. Laporan: Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 4
FRAGMENT

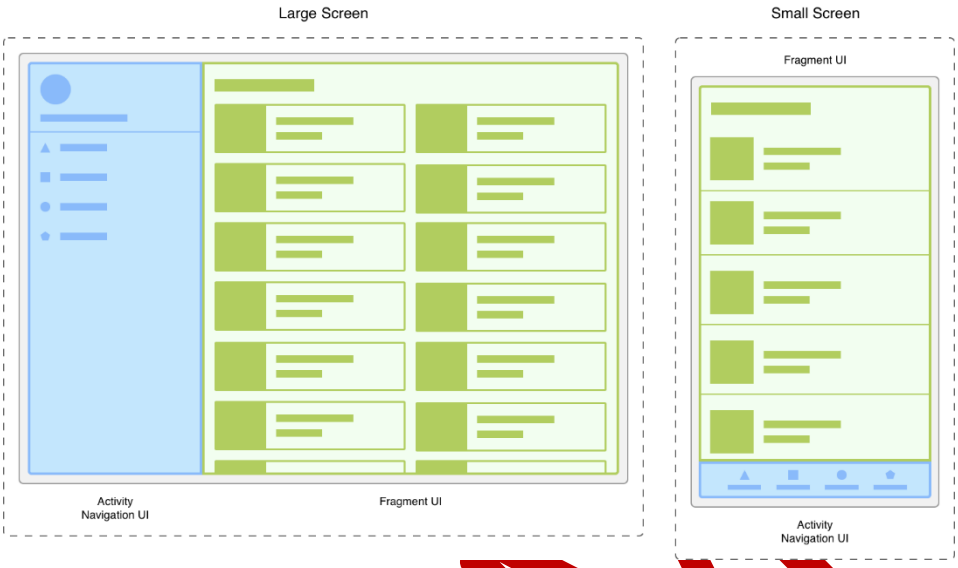
Dalam pengembangan aplikasi Android, kebutuhan untuk menciptakan antarmuka yang adaptif dan responsif menjadi semakin penting, terutama seiring dengan beragamnya ukuran dan orientasi layar perangkat. Pengembang dihadapkan pada tantangan untuk merancang aplikasi yang tetap intuitif dan konsisten, baik pada ponsel dengan layar kecil maupun tablet atau perangkat lainnya dengan layar lebih besar.

Keberadaan *Fragment* sangat penting dalam menciptakan aplikasi yang responsif dan adaptif terhadap variasi ukuran layar. *Fragment* memungkinkan *Modularitas*, fleksibilitas, dan penggunaan kembali antarmuka aplikasi yang efektif. *Fragment* juga mendukung pengelolaan elemen navigasi dan konten secara terpisah, sehingga memungkinkan navigasi yang lebih rapi serta manajemen tampilan yang lebih efisien. Selain itu, *Fragment* membantu dalam optimalisasi kinerja aplikasi dengan meminimalisir penggunaan memori dan konsumsi daya, menjadikannya pilihan penting dalam desain aplikasi yang efisien. Pada bab ini, akan dibahas lebih lanjut menganai *Fragment*.

A. *Fragment*

Fragment adalah bagian dari antarmuka pengguna (UI) dalam aplikasi Android yang dapat digunakan kembali dan dikelola secara independen. *Fragment* ini menentukan dan mengelola tata letaknya sendiri, memiliki siklus proses yang terpisah, dan mampu menangani peristiwa inputnya sendiri. Meskipun demikian, *Fragment* tidak dapat berfungsi sebagai komponen mandiri. *Fragment* harus dihosting oleh sebuah *Activity* atau *Fragment* lain yang mendukungnya, dan hierarki tampilan dari *Fragment* tersebut akan dilampirkan ke hierarki tampilan dari *Activity* atau *Fragment* host-nya.Ketik konten Sub Bab disini

DUMMY



Gambar 4. Perbandingan Tampilan UI: Layar Besar dengan Panel Navigasi vs Layar Kecil dengan Menu Navigasi Bawah
Sumber: https://developer.android.com/

DUMMY

Dalam konteks ini, *Fragment* bertindak sebagai "sub- *Activity*" yang memungkinkan pengembang untuk memecah antarmuka aplikasi menjadi bagian-bagian yang lebih kecil dan fleksibel. Dengan *Fragment*, antarmuka pengguna dapat dibangun dalam blok-blok terpisah yang memudahkan pengelolaan dan pemanfaatan kembali (reusability) di berbagai bagian aplikasi. Karena *Fragment* bisa diatur dan dimodifikasi secara independen, *Fragment* memberi fleksibilitas pada pengembang dalam menampilkan UI yang responsif, khususnya pada perangkat dengan variasi ukuran layar.

Penggunaan *Fragment* dalam pengembangan aplikasi Android memiliki beberapa tujuan utama, yang menjadikannya pilihan penting dalam menciptakan aplikasi yang fleksibel dan ramah pengguna:

1. ***Modularitas* dan Penggunaan Kembali**

   Fragment memungkinkan *Modularitas* dan penggunaan kembali antarmuka dalam aplikasi Android dengan membagi UI menjadi bagian-bagian yang lebih kecil dan terfokus. Setiap *Fragment* dapat mengelola dan menampilkan UI dari area tertentu, sementara aktivitas dapat digunakan untuk elemen global seperti navigasi. Pada perangkat besar, seperti tablet, *Fragment* dapat menampilkan daftar di satu panel dan detail di panel lainnya. Sementara itu, pada layar kecil, aplikasi bisa menggunakan menu navigasi bawah untuk menyederhanakan

tampilan. Dengan pendekatan ini, aktivitas bertugas mengelola elemen global, sementara *Fragment* menangani tata letak konten.

2. **Antarmuka yang Fleksibel dan Adaptif**

   Fragment mendukung antarmuka pengguna yang adaptif dan fleksibel, memungkinkan aplikasi untuk menyesuaikan tampilan pada perangkat dengan berbagai ukuran layar. Pada perangkat besar, *Fragment* dapat disusun dalam tata letak grid untuk menampilkan lebih banyak informasi, sementara pada perangkat kecil, tata letak bisa lebih sederhana atau menggunakan menu navigasi minimalis. Ini membantu menciptakan aplikasi responsif yang sesuai dengan berbagai ukuran layar.

3. **Pemisahan Navigasi dan Konten**

   Fragment memudahkan pemisahan elemen navigasi dan konten dalam aplikasi yang kompleks, seperti aplikasi belanja atau berita. Aktivitas mengelola navigasi global, sementara *Fragment* fokus pada tampilan konten spesifik. Ini memudahkan pengelolaan aplikasi karena *Fragment* dapat ditambah atau diganti tanpa mengubah aktivitas utama.

4. **Kinerja dan Efisiensi**

   Fragment memungkinkan aplikasi mengelola komponen UI secara efisien, di mana aktivitas hanya perlu memuat *Fragment* yang aktif. Dengan cara ini, beban memori berkurang dan konsumsi daya dapat dioptimalkan, terutama pada perangkat dengan sumber daya terbatas.

B. Kelas Dasar *Fragment*

Fragment dalam Android adalah komponen modular yang mewakili bagian dari antarmuka pengguna dalam suatu *Activity*. *Fragment* dibuat dengan menggunakan kelas dasar *Fragment*, yang berfungsi sebagai pondasi bagi setiap *Fragment* yang akan dibuat. Dalam pengembangan aplikasi Android, *Fragment* memungkinkan bagian antarmuka yang kompleks untuk dibagi menjadi bagian-bagian yang lebih kecil dan mandiri, sehingga dapat dikelola secara lebih efisien.

Android juga menyediakan kelas-kelas turunan khusus dari *Fragment*, seperti:

1. **Dialog*Fragment***

   Digunakan untuk menampilkan dialog yang interaktif dan melayang di atas antarmuka aplikasi. Dialog*Fragment* cocok untuk

DUMMY

menampilkan pesan singkat, konfirmasi, atau pilihan-pilihan sederhana kepada pengguna. Keuntungannya dibandingkan dialog biasa adalah bahwa Dialog*Fragment* terintegrasi dalam siklus hidup *Fragment*, sehingga lebih mudah dikelola dan dipertahankan.

2. ***List Fragment***

   Fragment khusus yang menyederhanakan pengelolaan daftar tampilan (*list view*). List*Fragment* dirancang untuk memudahkan pembuatan dan pengelolaan daftar data yang panjang, sehingga sering digunakan dalam aplikasi yang memerlukan tampilan daftar, seperti daftar produk, daftar kontak, atau menu pilihan.

3. ***Preference Fragment***

   Fragment ini digunakan untuk menampilkan preferensi atau pengaturan dalam format yang mudah dibaca oleh pengguna. Preference *Fragment* berguna untuk mengelola pengaturan aplikasi yang melibatkan input dari pengguna, seperti preferensi tampilan, notifikasi, atau opsi konfigurasi lainnya.

   Pada pengembangan aplikasi Android, *Fragment* digunakan untuk menciptakan antarmuka yang modular dan fleksibel. *Fragment* dapat ditambahkan ke dalam *Activity* baik secara dinamis melalui kode atau secara statis melalui XML. Berikut ini adalah panduan formal untuk membuat dan menambahkan kelas *Fragment* ke dalam *Activity*.

   a. Membuat Kelas Dasar *Fragment*

   Untuk membuat kelas dasar *Fragment*, perluasan dari *Fragment* dilakukan dengan membuat kelas Java atau Kotlin yang menurunkan kelas *Fragment*. Kelas ini biasanya berisi metode untuk mengelola siklus hidup *Fragment* serta untuk mendefinisikan antarmuka pengguna *Fragment*.

```
// Membuat kelas dasar Fragment di Kotlin
class ExampleFragment : Fragment() {
    override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
    ): View? {
    // Menghubungkan layout Fragment dengan file XML
    return
     inflater.inflate(R.layout.Fragment_example,
     container, false)
    }
}
```

DUMMY

b. Membuat Kelas Turunan *Fragment*

Android menyediakan kelas turunan *Fragment* khusus yang memiliki fungsi spesifik, antara lain:

1) Dialog *Fragment*

Untuk membuat dialog, buat kelas dengan menurunkan Dialog *Fragment*. Kelas ini berguna untuk menampilkan dialog yang diintegrasikan dengan siklus hidup *Fragment*.

```
class ExampleDialogFragment : DialogFragment() {
 override fun onCreateDialog(savedInstanceState:
Bundle?): Dialog {
 return AlertDialog. Builder(requireContext())
  .setTitle("Contoh Dialog")
  .setMessage("Ini adalah pesan dialog")
  .setPositiveButton("OK", null)
  .create()
 }
}
```

2) List *Fragment*

Untuk menampilkan daftar data, buat kelas dengan menurunkan List *Fragment*. Kelas ini menyederhanakan pengelolaan tampilan daftar dalam *Fragment*.

```
class ExampleListFragment : ListFragment() {
 override fun onActivityCreated(savedInstanceState:
Bundle?) {
 super.onActivityCreated(savedInstanceState)
 listAdapter = ArrayAdapter(
  requireContext(),
  android.R.layout.simple_list_item_1,
  arrayOf("Item 1", "Item 2", "Item 3")
 )
 }
}
```

3) Preference *Fragment*

Untuk menampilkan preferensi, buat kelas dengan menurunkan Preference*Fragment*. Ini cocok untuk menampilkan dan mengelola pengaturan aplikasi dalam bentuk daftar preferensi.

```
class ExamplePreferenceFragment :
PreferenceFragmentCompat() {
 override fun
onCreatePreferences(savedInstanceState: Bundle?,
rootKey: String?) {
 setPreferencesFrom resource (R.xml.preferences,
rootKey)
 }
```

DRAFT

```
}
```

c. Menambahkan *Fragment* Melalui Kode (Runtime)

Menambahkan *Fragment* secara dinamis memungkinkan *Fragment* untuk diubah atau dihapus saat aplikasi berjalan. Untuk melakukan ini, gunakan *Fragment*Manager dan *Fragment*Transaction untuk menambahkan, mengganti, atau menghapus *Fragment* dari *Activity*.

```
// Menambahkan Fragment secara dinamis dalam Activity
di Kotlin

class Main Activity : AppCompat Activity() {

 override fun onCreate(savedInstanceState: Bundle?) {

 super.onCreate(savedInstanceState)

 setContentView(R.layout. Activity_main)


 // Membuat instance Fragment

 val Fragment = ExampleFragment()


 // Menambahkan Fragment ke dalam layout container

 supportFragmentManager.beginTransaction()

  .add(R.id.Fragment_container, Fragment) //
R.id.Fragment_container adalah ID dari container
layout

  .commit()

 }

}
```

Pada contoh di atas, android:name merujuk pada nama kelas lengkap dari *Fragment* yang akan ditampilkan.

d. Menambahkan *Fragment* Melalui XML

Fragment dapat dimasukkan secara langsung ke dalam layout XML *Activity*, sehingga *Fragment* akan dimuat bersama dengan layout *Activity* saat dijalankan. Ini berguna untuk *Fragment* yang akan selalu ditampilkan pada layout tersebut.

```
<!-- Contoh menambahkan Fragment dalam layout XML -->
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/andr
```

```
oid"
android:layout_width="match_parent"
android:layout_height="match_parent">

<Fragment
 android:id="@+id/example_Fragment"
 android:name="com.example.app.ExampleFragment"
 android:layout_width="match_parent"
 android:layout_height="match_parent" />
</FrameLayout>
```

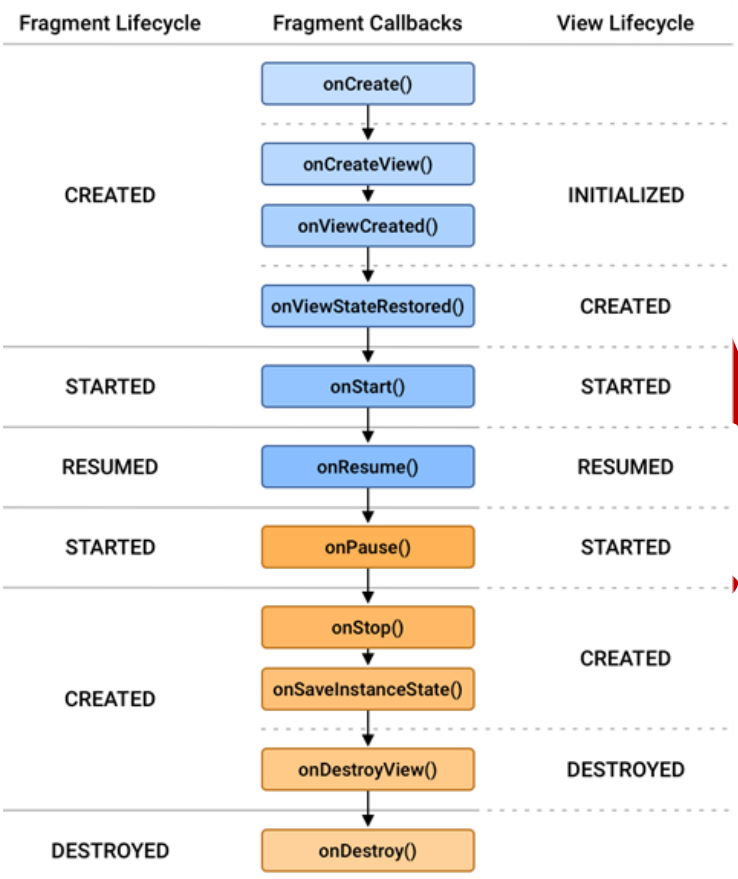
Dalam contoh ini, `R.id.Fragment_container` merujuk ke ID dari container layout (seperti `FrameLayout`) tempat *Fragment* akan ditambahkan. `Fragment` ditambahkan menggunakan `add()` pada `FragmentTransaction`, yang kemudian dijalankan dengan `commit()`.

Dengan menggunakan *Fragment*, pengembang dapat menciptakan antarmuka yang modular, fleksibel, dan mudah diadaptasi. Proses menambahkan *Fragment* baik secara statis melalui XML atau secara dinamis melalui kode memberikan keleluasaan dalam mengelola tampilan aplikasi.

C. Siklus Hidup *Fragment Lifecycle*

Fragment adalah komponen modular dalam Android yang memiliki siklus hidup tersendiri, yang berbeda namun sejalan dengan siklus hidup *Activity* yang menjadi host-nya. Siklus hidup *Fragment* sangat penting untuk memahami bagaimana *Fragment* berinteraksi dengan *Activity*, serta kapan dan bagaimana melakukan inisialisasi, penanganan data, dan pembersihan sumber daya. Pemahaman yang baik mengenai siklus hidup *Fragment* sangat penting karena *Fragment* berperan dalam pengelolaan antarmuka aplikasi yang fleksibel dan efisien.

DUMMY



Gambar 5. Lifecycle Fragment
Sumber : developer.android.com

Berikut ini adalah penjelasan mendalam mengenai setiap metode utama dalam siklus hidup *Fragment*:

1. *onAttach*()

Metode onAttach() dipanggil ketika *Fragment* pertama kali melekat pada suatu *Activity*. Tahap ini adalah momen awal ketika Fragment mendapatkan akses ke konteks *Activity* yang menjadi host-nya. Melalui metode ini, *Fragment* dapat berkomunikasi dengan *Activity* serta mendapatkan akses ke *resource* yang mungkin dibutuhkan. Ini adalah titik awal di mana *Fragment* mengkonfirmasi bahwa *Activity* yang menjadi host-nya tersedia dan dapat digunakan untuk interaksi lebih lanjut.

```
override fun onAttach(context: Context) {
 super.onAttach(context)
 // Inisialisasi atau akses ke Activity dilakukan di sini
```

DUM

```
}
```

2. *onCreate*()

Metode onCreate() adalah tahap pertama inisialisasi dalam siklus hidup *Fragment*. Pada tahap ini, *Fragment* belum memiliki tampilan yang terkait dengannya, sehingga kode yang tidak memerlukan akses ke tampilan dapat ditempatkan di sini. onCreate() sering digunakan untuk menginisialisasi data penting atau konfigurasi awal yang dibutuhkan *Fragment*. Ini mirip dengan metode onCreate() pada *Activity*.

```
override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 // Lakukan inisialisasi data atau konfigurasi yang
diperlukan
}
```

3. *onCreateView*()

Metode onCreateView() dipanggil untuk menciptakan antarmuka pengguna dari *Fragment*. Pada tahap ini, *Fragment* harus membuat tampilan utama yang akan ditampilkan oleh *Activity*. Biasanya, layout *Fragment* didefinisikan dalam file XML, dan metode ini mengembalikan tampilan (view) yang dibuat berdasarkan layout tersebut. onCreateView() bertanggung jawab penuh atas pembuatan tampilan *Fragment* dan pengaturan elemen-elemen UI awal.

```
override fun onCreateView(
 inflater: LayoutInflater, container: ViewGroup?,
 savedInstanceState: Bundle?
): View? {
 return inflater.inflate(R.layout.Fragment_example,
container, false)
}
```

4. *onStart*()

Metode onStart() dipanggil ketika *Fragment* mulai menjadi terlihat bagi pengguna. Pada tahap ini, *Fragment* sudah berada dalam hirarki tampilan dan tampak pada layar, tetapi belum dalam kondisi aktif penuh. Biasanya, metode ini digunakan untuk mempersiapkan antarmuka pengguna atau menyiapkan *resource* yang akan digunakan saat *Fragment* berada dalam kondisi terlihat.

```
override fun onStart() {
 super.onStart()
```

```
 // Persiapan tampilan atau inisialisasi yang diperlukan
saat Fragment terlihat
}
```

5. *onResume*()

Metode onResume() adalah tahap ketika *Fragment* sudah aktif dan siap berinteraksi dengan pengguna. Pada tahap ini, *Fragment* dalam kondisi aktif penuh, dan segala interaksi pengguna dapat ditangani di sini. Misalnya, jika *Fragment* membutuhkan input pengguna, animasi, atau data yang diperbarui secara real-time, aktivitas tersebut dapat dimulai atau dilanjutkan di dalam metode ini.

```
override fun onResume() {
 super.onResume()
 // Mulai atau lanjutkan aktivitas yang memerlukan
interaksi pengguna
}
```

6. *onPause*()

Metode onPause() dipanggil ketika *Fragment* tidak lagi aktif atau dalam kondisi jeda (pause). Biasanya, onPause() digunakan untuk menghentikan aktivitas yang tidak boleh berjalan di background atau yang memerlukan perhatian penuh dari pengguna, seperti animasi, pemutar video, atau proses yang sensitif terhadap daya.

```
override fun onPause() {
 super.onPause()
 // Hentikan atau jedakan aktivitas yang memerlukan fokus
pengguna
}
```

7. *onStop*()

Metode onStop() dipanggil saat *Fragment* tidak lagi terlihat oleh pengguna. Pada tahap ini, *Fragment* dalam kondisi non-aktif, namun tetap terpasang pada *Activity*. onStop() biasanya digunakan untuk menghentikan operasi berat atau membebaskan *resource* yang tidak dibutuhkan saat *Fragment* tidak terlihat, misalnya menghentikan pembaruan data yang tidak diperlukan atau menghentikan animasi yang berjalan.

```
override fun onStop() {
 super.onStop()
 // Hentikan aktivitas atau bebaskan resource yang tidak
diperlukan
}
```

8. *onDestroyView*()

Metode onDestroy() adalah tahap akhir dari siklus hidup *Fragment*. Pada tahap ini, *Fragment* akan dihapus sepenuhnya dari memori. Biasanya, onDestroy() digunakan untuk melakukan pembersihan akhir dari *resource* yang tidak lagi diperlukan, seperti observer atau listener yang mungkin terdaftar di *Fragment*.

```
override fun onDestroy() {
 super.onDestroy()
 // Bersihkan resource atau listener yang tidak lagi
diperlukan
}
```

9. *onDestroy*()

Metode onDestroy() adalah tahap akhir dari siklus hidup *Fragment*. Pada tahap ini, *Fragment* akan dihapus sepenuhnya dari memori. Biasanya, onDestroy() digunakan untuk melakukan pembersihan akhir dari *resource* yang tidak lagi diperlukan, seperti observer atau listener yang mungkin terdaftar di *Fragment*.

```
override fun onDestroy() {
 super.onDestroy()
 // Bersihkan resource atau listener yang tidak lagi
diperlukan
}
```

10. *onDetach*()

Metode `onDetach()` adalah langkah terakhir dalam siklus hidup *Fragment*, di mana *Fragment* dilepaskan dari *Activity* yang menjadi host-nya. Setelah `onDetach()` dipanggil, *Fragment* tidak lagi memiliki akses ke *Activity*. Metode ini digunakan untuk membersihkan referensi terakhir dan memastikan bahwa *Fragment* tidak lagi berkomunikasi dengan *Activity*.

```
override fun onDetach() {
 super.onDetach()
 // Bersihkan referensi terakhir ke Activity
}
```

D. Implementasi *Fragment*

Langkah - Langkah Implementasi Aplikasi:

1. Membuat *Activity* File

   Langkah 1.1: Buka kembali project

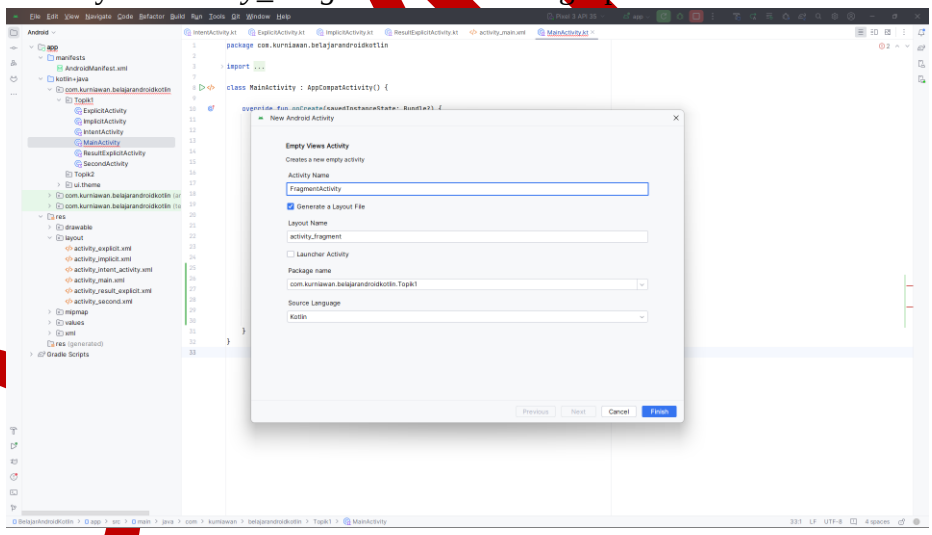
Belajar Kotlin yang telah di buat pada Job Sheet sebelumnya

Langkah 1.2: Perbarui kode pada Main *Activity*.kt

MainActivity.kt

```kotlin
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val buttonStart = findViewById<Button>(R.id.button_start)
        buttonStart.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, SecondActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan3 = findViewById<Button>(R.id.button_pertemuan3)
        buttonPertemuan3.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, IntentActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan4 = findViewById<Button>(R.id.button_pertemuan4)
        buttonPertemuan4.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, FragmentActivity::class.java) // Pastikan menggunakan path yang lengkap
            startActivity(intent)
        }
    }
}
```

Langkah 1.3: Buat *Activity* baru dengan nama *Fragment Activity*.kt dan Layout *Activity_Fragment*.xml dan lengkapi kode :



MainActivity.kt FragmentActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.google.android.material.bottomnavigation.BottomNavigationView
import com.kurniawan.belajarandroidkotlin.HomeFragment
import com.kurniawan.belajarandroidkotlin.InfoFragment
import com.kurniawan.belajarandroidkotlin.ProfileFragment
import com.kurniawan.belajarandroidkotlin.R

class FragmentActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_fragment)

        // Load the default fragment (HomeFragment)
        if (savedInstanceState == null) {
            supportFragmentManager.beginTransaction()
                .replace(R.id.fragment_container, HomeFragment())
                .commit()
        }

        val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom_navigation)

        // Set listener for navigation item selection
        bottomNavigationView.setOnNavigationItemSelectedListener { item ->
            val selectedFragment = when (item.itemId) {
                R.id.nav_home -> HomeFragment()
                R.id.nav_profile -> ProfileFragment()
                R.id.nav_info -> InfoFragment()
                else -> null
            }

            // Replace the fragment if it's not null
            selectedFragment?.let { it: Fragment
                supportFragmentManager.beginTransaction()
                    .replace(R.id.fragment_container, it as Fragment)
                    .commit()
            }
            true ^setOnNavigationItemSelectedListener
        }
    }
}
```

Langkah 1.4: Buat *Activity* baru dengan nama Home*Fragment*.kt dan isikan contoh kode berikut :

HomeFragment.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class HomeFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.activity_home_fragment, container, attachToRoot: false)
    }
}
```

Langkah 1.5: Buat *Activity* baru dengan nama Profile*Fragment*.kt dan isikan contoh kode berikut :

HomeFragment.kt ProfileFragment.kt

```
package com.kurniawan.belajarandroidkotlin // Pastikan menggunakan package yang sesuai

import ...

class ProfileFragment : Fragment() { // Ubah dari AppCompatActivity ke Fragment
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.activity_profile_fragment, container, attachToRoot: false) // Sesuaikan layout
    }
}
```

Langkah 1.6: Buat *Activity* baru dengan nama Info*Fragment*.kt dan isikan contoh kode berikut :

HomeFragment.kt ProfileFragment.kt InfoFragment.kt

```
package com.kurniawan.belajarandroidkotlin

import ...

class InfoFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.activity_info_fragment, container, attachToRoot: false)
    }
}
```

2. Menambahkan Komponen UI pada Layout

Langkah 2.1: Buat Directory baru dengan nama Menu dan tambahkan file baru dengan nama bottton_nav_menu.xml isikan contoh kode berikut :

buttom_nav_menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/nav_home"
        android:icon="@drawable/home"
        android:title="Home" />
    <item
        android:id="@+id/nav_profile"
        android:icon="@drawable/profile"
        android:title="Profile"/>
    <item
        android:id="@+id/nav_info"
        android:icon="@drawable/info"
        android:title="Info"/>
</menu>
```

Langkah 2.2: Download File icon home, profile dan info di https://www.svgrepo.com/ dan tambahkan ke directory drawable dengan cara klik kanan > New > Vector Asset. Ganti path file dan arahkan pada lokasi file icon yang sudah didownload.

Langkah 2.3: Tambahkan pada *Activity*_main.xml script berikut

```
<LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:gravity="center">

 <Button
 android:id="@+id/button_pertemuan4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Fragment Activity" />
</LinearLayout>
```

Langkah 2.4: Isikan Layout *Activity_Fragment* dengan script berikut:

```
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".HomeFragment">

 <FrameLayout
 android:id="@+id/Fragment_container"
 android:layout_width="0dp"
 android:layout_height="0dp"
 app:layout_constraintTop_toTopOf="parent"

app:layout_constraintBottom_toTopOf="@+id/bottom_naviga
tion"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent" />


<com.google.android.material.bottomnavigation.BottomNav
igationView
 android:id="@+id/bottom_navigation"
 android:layout_width="match_parent"
```

```
 android:layout_height="wrap_content"

app:itemBackground="@color/design_default_color_primary
"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:menu="@menu/buttom_nav_menu" />

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

Langkah 2.5: Isikan Layout *Fragment*_home dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Ini Adalah Home Fragment"
 android:textSize="24dp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"></TextView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Langkah 2.6: Isikan Layout *Fragment*_profile dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Ini Adalah Profile Fragment"
```

```
 android:textSize="24dp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"></TextView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Langkah 2.7: Isikan Layout *Fragment*_info dengan script berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/andro
id"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent">
 <TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Ini Adalah Info Fragment"
 android:textSize="24dp"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent"></TextView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Menjalankan dan Menguji Aplikasi

   Langkah 3.1: Klik Run atau tekan Shift + F10 untuk menjalankan aplikasi.

   Langkah 3.2: Pilih emulator atau perangkat fisik untuk menjalankan aplikasi.

DU

4. Hasil Pengujian

Gambar 6. Hasil Implementasi *Fragment*

E. Tugas

1. **Modifikasi:** Tambahkan Input User *Activity* dan Profile *Activity*.
2. **Eksplorasi:**
   a. Pada InputUser *Activity*, siapkan dua input field untuk memasukkan nama dan umur.
   b. Tambahkan sebuah tombol "Kirim". Ketika tombol ini ditekan, aplikasi akan mengirim data nama dan umur tersebut ke Profile *Activity* menggunakan Intent.
   c. Pada Profile *Activity* tampilkan nama dan umur yang diterima dari InputUser *Activity* .
   d. Gunakan TextView di Profile *Activity* untuk menampilkan hasilnya.
3. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 5
LAYOUT

Layout merupakan struktur yang menentukan bagaimana elemen-elemen visual, seperti teks, gambar, dan tombol, ditempatkan dan diatur dalam layar aplikasi. Pengetahuan tentang layout menjadi fundamental bagi para pengembang, karena desain yang baik tidak hanya memperindah tampilan tetapi juga meningkatkan pengalaman pengguna dengan membuat navigasi lebih mudah dan responsif. Bab ini akan mengulas berbagai jenis layout yang umum digunakan, seperti LinearLayout, RelativeLayout, dan ConstraintLayout, serta memberikan panduan praktis tentang kapan dan bagaimana memilih jenis layout yang tepat sesuai kebutuhan aplikasi. Dengan memahami prinsip-prinsip dasar layout, pembaca akan memiliki pondasi kuat untuk menciptakan aplikasi Android yang user-friendly dan estetik.

A. Linear Layout

Linear Layout adalah salah satu jenis layout dalam Android yang digunakan untuk mengatur tampilan elemen-elemen secara linear, baik secara vertikal maupun horizontal. Elemen-elemen ditempatkan satu per satu sesuai dengan arah yang ditentukan. Jika diatur secara vertikal, setiap elemen akan berada di bawah elemen sebelumnya, sedangkan secara horizontal, elemen-elemen akan ditempatkan di samping elemen sebelumnya.Ketik konten Sub Bab disini

1. **Orientation:** Atribut ini menentukan apakah elemen-elemen dalam layout akan diatur secara vertikal (vertical) atau horizontal (horizontal).
2. **Weight:** Properti layout_weight digunakan untuk menentukan bagaimana ruang yang tersisa dibagi di antara elemen-elemen dalam layout.
3. **Gravity dan Layout Gravity:** Atribut gravity menentukan bagaimana konten ditempatkan di dalam elemen, sedangkan layout_gravity mengatur posisi elemen itu sendiri dalam layout.

Keunggulan Linear Layout:

1. Mudah Dipahami dan Diimplementasikan
   Linear Layout sangat mudah dipelajari dan digunakan, terutama untuk menata elemen secara berurutan, baik secara vertikal maupun horizontal.

DUMMY

2. Pengaturan Ruang yang Dinamis dengan layout_weight
   Linear Layout memiliki atribut layout_weight, yang memungkinkan kita mengatur ruang antar elemen secara proporsional. Ini berguna untuk membuat tampilan lebih responsif pada berbagai ukuran layar.
3. Cocok untuk Desain Sederhana
   Linear Layout adalah pilihan yang tepat untuk desain yang sederhana tanpa banyak elemen atau kebutuhan pengaturan yang rumit.

Kelemahan Linear Layout:

1. Kurang Efisien untuk Desain yang Kompleks
   Linear Layout kurang cocok untuk desain yang kompleks karena elemen hanya bisa ditempatkan dalam satu arah (vertikal atau horizontal), sehingga sulit digunakan untuk struktur tampilan yang rumit.
2. Kinerja Menurun pada Banyak Elemen
   Jika ada terlalu banyak elemen, khususnya dalam satu arah, Linear Layout bisa menyebabkan masalah kinerja karena membutuhkan waktu lebih lama untuk merender semua elemen.
3. Tidak Mendukung Penataan Bebas
   Linear Layout hanya menata elemen secara berurutan, sehingga membatasi fleksibilitas penataan elemen. Untuk tampilan yang membutuhkan penempatan lebih bebas, Linear Layout kurang ideal.

B. Relative Layout

Relative Layout adalah layout yang mengatur elemen berdasarkan posisi relatifnya terhadap elemen-elemen lain atau terhadap parent-nya. Layout ini memberi fleksibilitas lebih dibandingkan Linear Layout karena memungkinkan kita untuk menentukan posisi komponen relatif terhadap komponen lain (misalnya, di bawah, di atas, di samping) maupun relatif terhadap boundary layout itu sendiri.

1. **Alignment**: Elemen-elemen dapat diatur relatif terhadap elemen lain menggunakan atribut seperti layout_toRightOf, layout_below, layout_alignParentTop, dan lain-lain.
2. **Parent Constraints**: Elemen juga dapat diposisikan relatif terhadap tepi layout menggunakan atribut seperti layout_alignParentLeft atau layout_alignParentBottom.

Keunggulan Relative Layout:

DUMMY

1. Fleksibel untuk Penataan yang Kompleks
   Relative Layout memudahkan kita untuk mengatur posisi elemen UI di atas, di bawah, atau di samping elemen lainnya. Ini berguna jika kita ingin tampilan yang lebih dinamis dan fleksibel dibandingkan layout sederhana.
2. Mengurangi Penggunaan Nested Layout
   Karena bisa menempatkan elemen UI relatif terhadap satu sama lain, Relative Layout membantu menghindari penggunaan layout bertumpuk (nested layout), yang artinya tidak perlu menumpuk banyak layout di dalam satu sama lain. Ini membuat tampilan lebih rapi dan bisa meningkatkan kinerja aplikasi.
3. Cocok untuk Tampilan Dinamis
   Relative Layout ideal jika kita ingin elemen-elemen UI yang saling terhubung dan mudah diubah posisinya berdasarkan kondisi tertentu. Misalnya, kita bisa menempatkan elemen "di bawah" elemen lain dan mengubah urutannya sesuai kebutuhan.

Kelemahan Relative Layout:

1. Sulit Diatur Jika Elemen Terlalu Banyak
   Jika elemen UI dalam Relative Layout terlalu banyak, kita akan kesulitan mengatur posisi relatif antar elemen, karena semuanya harus saling bergantung. Akibatnya, tata letak bisa menjadi sulit dibaca, dikelola, dan dimodifikasi.
2. Kinerja Lambat pada Tampilan yang Rumit
   Relative Layout membutuhkan waktu lebih lama untuk menghitung posisi setiap elemen, terutama pada tampilan yang sangat kompleks. Hal ini bisa membuat aplikasi berjalan lebih lambat, terutama jika ada banyak elemen yang harus dihitung posisinya satu per satu.
3. Cukup Sulit Dipahami bagi Pemula
   Karena posisi setiap elemen bergantung pada posisi elemen lain, Relative Layout bisa membingungkan bagi pemula. Dibutuhkan pemahaman yang baik tentang penataan elemen dan hubungan antar elemen dalam tampilan.Ketik konten Sub Bab disini

C. Constraint Layout

Constraint Layout adalah layout yang lebih kuat dan fleksibel dibandingkan Linear Layout dan Relative Layout, dirancang untuk menangani antarmuka yang lebih kompleks dengan cara yang lebih

DUMMY

efisien. Layout ini menggunakan sistem constraint yang mengatur posisi elemen berdasarkan constraint ke elemen lain atau ke parent-nya.

1. **Constraint:** Setiap elemen dalam layout harus memiliki constraint yang menghubungkannya dengan elemen lain, misalnya, constraint ke sisi lain atau ke tepi layout.
2. **Bias:** Atribut bias memungkinkan penempatan elemen dengan posisi yang lebih presisi antara dua constraint.
3. **Chains:** Elemen-elemen bisa dikelompokkan dalam rantai (chain) untuk mengontrol distribusi ruang antar elemen.
4. **Guideline dan Barrier:** Guideline digunakan untuk mengatur posisi secara dinamis berdasarkan persentase, sementara Barrier memungkinkan pengaturan dinamis yang lebih fleksibel berdasarkan elemen lain.

Keunggulan ConstraintLayout

1. Fleksibilitas Tinggi untuk Penataan Elemen
   ConstraintLayout memungkinkan kita menata elemen-elemen UI dengan lebih bebas, baik secara vertikal maupun horizontal. Elemen dapat ditempatkan di mana saja dalam layout dengan menghubungkannya ke elemen lain atau tepi layout.
2. Mengurangi Kebutuhan Nested Layout
   ConstraintLayout memungkinkan kita menempatkan elemen dengan berbagai hubungan tanpa perlu menggunakan nested layout (layout bertumpuk). Ini membuat tampilan lebih sederhana dan meningkatkan kinerja aplikasi.
3. Ideal untuk Desain Responsif
   ConstraintLayout memungkinkan pengaturan elemen yang lebih responsif di berbagai ukuran layar, sehingga tampilan tetap rapi di berbagai perangkat tanpa perlu banyak pengaturan tambahan.

Kelemahan Constraint Layout

1. Cukup Kompleks untuk Pemula
   Karena fleksibilitasnya yang tinggi, ConstraintLayout membutuhkan pemahaman yang lebih mendalam tentang pengaturan constraint atau hubungan antar elemen, sehingga mungkin sedikit rumit bagi pemula.
2. Pengaturan Constraint yang Membingungkan
   Menghubungkan elemen satu sama lain dengan constraint bisa menjadi rumit, terutama pada tampilan yang sangat kompleks, karena setiap

DUMMY

elemen harus memiliki hubungan yang jelas dengan elemen lain atau tepi layout.

3. Mengatur Desain Sederhana Bisa Terasa Berlebihan
   Untuk tampilan yang sangat sederhana, penggunaan ConstraintLayout bisa terasa terlalu rumit dibandingkan Linear Layout, yang lebih sederhana dan lebih mudah diatur jika hanya membutuhkan penataan sederhana.Ketik konten Sub Bab disini

D. Implementasi Layout

Langkah - Langkah implementasi pada aplikasi

1. **Membuat Menu Layout**

   Langkah 1.1: Buka kembali project Belajar Kotlin yang telah di buat pada bab sebelumnya.

   Langkah 1.2: Perbarui kode pada ***Main Activity.kt***

```
val buttonPertemuan4 = findViewById<Button>(R.id.button_pertemuan4)
buttonPertemuan4.setOnClickListener {
    val intent = Intent( packageContext: this, FragmentActivity::class.java)
    startActivity(intent)
}

val buttonPertemuan5 = findViewById<Button>(R.id.button_pertemuan5)
buttonPertemuan5.setOnClickListener {
    val intent = Intent( packageContext: this, LayoutActivity::class.java)
    startActivity(intent)
}
```

   Langkah 1.3: Buat *Activity* baru dengan nama ***Layout Activity.kt*** **dan** buatlah kode seperti di bawah berikut **:**

DUMMY

LayoutActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

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

        val buttonLinearlayout = findViewById<Button>(R.id.button_linearlayout)
        val buttonRelativelayout = findViewById<Button>(R.id.button_relativelayout)
        val buttonConstraintlayout = findViewById<Button>(R.id.button_constraintlayout)

        buttonLinearlayout.setOnClickListener {
            startActivity(Intent( packageContext: this, LinearLayoutActivity::class.java))
        }

        buttonRelativelayout.setOnClickListener {
            startActivity(Intent( packageContext: this, RelativeLayout::class.java))
        }

        buttonConstraintlayout.setOnClickListener {
            startActivity(Intent( packageContext: this, ConstraintLayout::class.java)) // Menghubungkan ke ConstraintLayou
        }
    }
}
```

Langkah 1.4: Buat *Activity* baru dengan nama ***Linearlayout Activity.kt*** dan isikan kode berikut:

LinearLayoutActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class LinearLayoutActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_linear_layout)
    }
}
```

Langkah 1.5: Buat *Activity* baru dengan nama ***Relativelayout.kt*** dan isikan kode berikut:

LinearLayoutActivity.kt RelativeLayout.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class RelativeLayout : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_relative_layout)
    }
}
```

Langkah 1.6: Buat *Activity* baru dengan nama ***Constraintlayout.kt*** dan isikan kode berikut:

LinearLayoutActivity.kt RelativeLayout.kt ConstraintLayout.kt

```kotlin
package com.kurniawan.belajarandroidkotlin

import ...

class ConstraintLayout : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_constraint_layout)
    }
}
```

2. **Menambahkan Komponen UI**

Langkah 2.1: Perbarui kode xml pada *Activity*_main.xml dengan menambahkan kode berikut:

```
<Button
 android:id="@+id/button3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Topik 3 - Layout" />
```

Langkah 2.2: Isikan kode xml ke ***Activity_layout*** dengan Root Tag LinearLayout seperti berikut:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_marginTop="24dp"
 android:orientation="vertical"
 android:padding="16dp">

 <!-- TextView untuk judul -->
 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginStart="10dp"
 android:layout_marginTop="10dp"
 android:gravity="center"
 android:text="@string/belajar_layout_android"
 android:textSize="15sp"
 android:textStyle="bold" />

 <!-- Linear Layout untuk Button Linear Layout -->
 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:gravity="center">

 <Button
  android:id="@+id/button_linearlayout"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/linear_layout" />
 </LinearLayout>

 <!-- Linear Layout untuk Button Relative Layout -->
 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp"
 android:gravity="center">

 <Button
  android:id="@+id/button_relativelayout"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
```

```
 android:text="@string/relative_layout" />
</LinearLayout>

<!-- Linear Layout untuk Button Constraint Layout -->
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
android:gravity="center">

<Button
 android:id="@+id/button_constraintlayout"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/constraint_layout" />
</LinearLayout>
</LinearLayout>
```

Langkah 2.3: Isikan kode xml ke ***Activity_linearlayout*** dengan Root Tag LinearLayout seperti berikut:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 android:layout_marginTop="24dp"
 android:padding="16dp">

 <!-- TextView with string resource -->
 <TextView
 android:id="@+id/textView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/linear_layout_ Activity"
 android:textSize="18sp"
 android:textColor="#000000"
 android:layout_marginBottom="16dp" />

 <!-- EditText with appropriate inputType -->
 <EditText
 android:id="@+id/editText"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:hint="@string/belajar_layout_android"
 android:inputType="textPersonName"
 android:layout_marginBottom="16dp" />
```

```
 <Button
 android:id="@+id/button"
 android:layout_width="match_parent"
 android:layout_height="48dp"
 android:text="@string/submit"
 android:textColor="#FFFFFF"
 android:backgroundTint="#6200EE"
 android:layout_marginBottom="16dp" />

 <!-- ImageView with contentDescription for accessibility
-->
 <ImageView
 android:id="@+id/imageView"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:src="@drawable/ic_launcher_foreground"
 android:contentDescription="@string/image_description"
 android:tint="#000000" />
</LinearLayout>
```

Langkah 2.4: Isikan kode xml ke ***Activity_relativelayout*** dengan Root Tag RelativeLayout seperti berikut:

```
<RelativeLayout

xmlns:android="http://schemas.android.com/apk/res/android
"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:padding="16dp">

 <TextView
 android:id="@+id/textViewRelative"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/relative_layout_ Activity"
 android:textSize="18sp"
 android:layout_alignParentTop="true"
 android:layout_centerHorizontal="true" />

 <Button
 android:id="@+id/buttonRelative"
 android:layout_width="wrap_content"
 android:layout_height="48dp"
 android:text="@string/submit"
 android:layout_below="@id/textViewRelative"
```

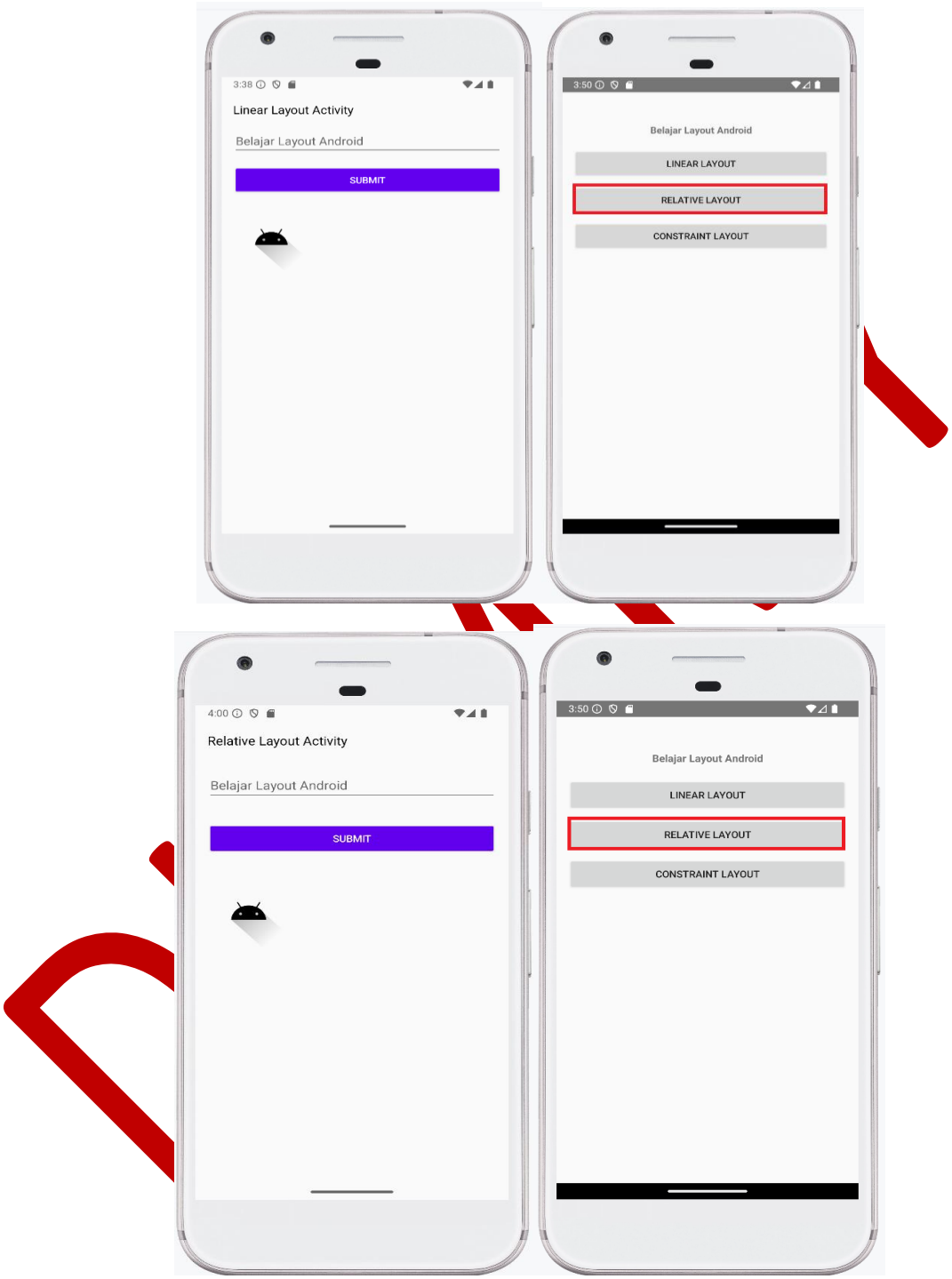
```
 android:layout_centerHorizontal="true"
 android:layout_marginTop="16dp" />

 <ImageView
 android:id="@+id/imageViewRelative"
 android:layout_width="48dp"
 android:layout_height="48dp"
 android:layout_below="@id/buttonRelative"
 android:layout_centerHorizontal="true"
 android:layout_marginTop="16dp"
 android:importantForAccessibility="no"
 android:contentDescription="@string/image_description"
 android:src="@drawable/ic_launcher_foreground"
 tools:ignore="ImageContrastCheck" />
</RelativeLayout>
```

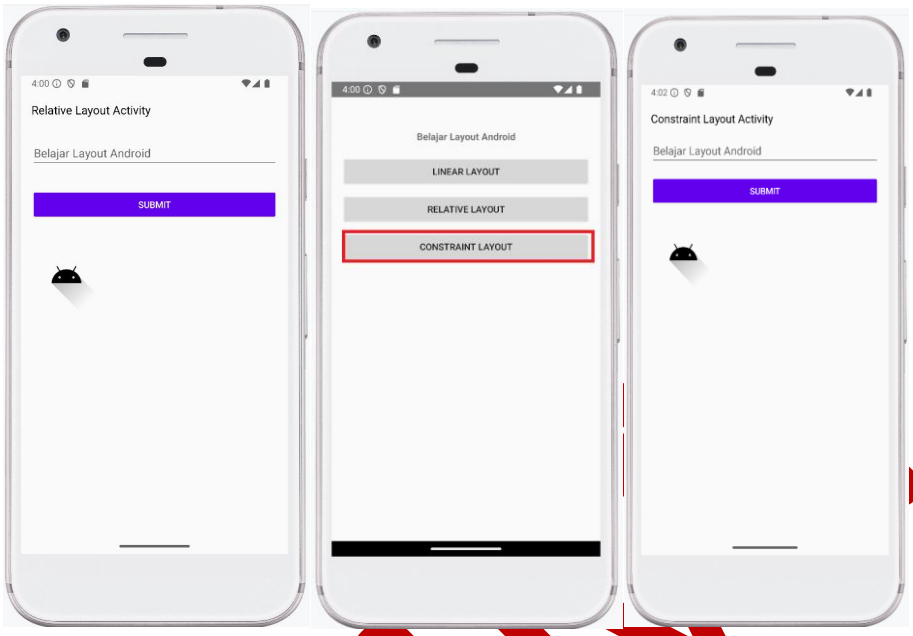
Langkah 2.5: Isikan kode xml ke ***Activity_constraintlayout*** dengan Root Tag Constraintlayout seperti berikut:

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:layout_marginTop="24dp"
 android:padding="16dp">

 <TextView
 android:id="@+id/textViewConstraint"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/constraint_layout_ Activity"
 android:textSize="18sp"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent" />

 <Button
 android:id="@+id/buttonConstraint"
 android:layout_width="wrap_content"
 android:layout_height="48dp"
 android:text="Submit"

app:layout_constraintTop_toBottomOf="@id/textViewConstrai
nt"
 app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="16dp"
    tools:ignore="HardcodedText" />

    <ImageView
    android:id="@+id/imageViewConstraint"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:importantForAccessibility="no"
    android:src="@drawable/ic_launcher_foreground"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"

   app:layout_constraintTop_toBottomOf="@id/buttonConstraint
   "
    tools:ignore="ImageContrastCheck" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

E. Hasil Pengujian


3:38
Linear Layout Activity
Belajar Layout Android
SUBMIT
3:50
Belajar Layout Android
LINEAR LAYOUT
RELATIVE LAYOUT
CONSTRAINT LAYOUT
4:00
Relative Layout Activity
Belajar Layout Android
SUBMIT
3:50
Belajar Layout Android
LINEAR LAYOUT
RELATIVE LAYOUT
CONSTRAINT LAYOUT



Gambar 7. Hasil Pengujian *Layout*

F. Tugas

1. **Modifikasi** : Setelah menyelesaikan setiap layout di atas, lakukan modifikasi berikut:
   a. Tambahkan warna latar belakang yang berbeda untuk setiap elemen layout.
   b. Gunakan padding dan margin untuk memberikan ruang antar elemen.
   c. Ubah properti teks seperti warna teks, ukuran font, dan gaya teks.
2. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 6
DIALOG, TOAST, SNACKBAR, ACTIONBAR, DAN MULTITHREADING

Pada bab ini, kita akan membahas berbagai komponen penting dalam pengembangan aplikasi Android, yaitu Dialog, Toast, Snackbar, ActionBar, dan Multithreading. Setiap komponen ini memainkan peran khusus dalam meningkatkan interaksi, fungsionalitas, dan performa aplikasi Android.

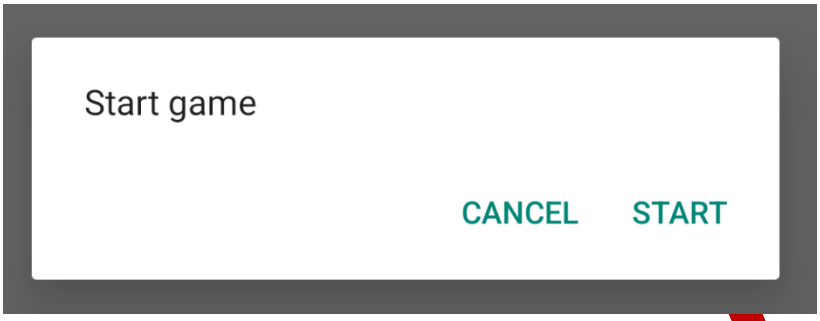
A. Dialog digunakan untuk menampilkan pesan atau pilihan kepada pengguna dalam bentuk jendela kecil.
B. Toast memberikan notifikasi singkat di layar tanpa mengganggu interaksi pengguna.
C. Snackbar menampilkan pesan dengan opsi tindakan yang dapat digunakan oleh pengguna.
D. ActionBar adalah bilah alat di bagian atas layar yang sering berisi tombol navigasi dan opsi menu.
E. Multithreading memungkinkan aplikasi untuk melakukan banyak tugas secara bersamaan, yang sangat berguna untuk menjaga aplikasi tetap responsif.

A. Dialog

Dialog adalah salah satu komponen antarmuka pengguna (UI) dalam aplikasi Android yang digunakan untuk berinteraksi dengan pengguna dalam konteks tertentu. Dialog pada dasarnya adalah elemen modal, yang berarti dialog akan muncul di atas konten aplikasi yang sedang berjalan, dan pengguna harus memberikan interaksi (seperti mengklik tombol atau memilih opsi) sebelum kembali ke layar utama aplikasi. Ini menjadikan dialog sangat efektif dalam memberikan konfirmasi, meminta masukan, atau menyampaikan informasi penting yang memerlukan perhatian segera dari pengguna.

Dalam pengembangan aplikasi Android, penggunaan dialog sangat penting karena memungkinkan pengembang untuk menciptakan pengalaman pengguna yang interaktif dan responsif. Dengan dialog, aplikasi dapat memberikan pemberitahuan atau meminta keputusan pengguna dalam situasi yang memerlukan perhatian khusus tanpa mengganggu alur aplikasi utama. Oleh karena itu, dialog sering digunakan untuk memunculkan pesan penting atau untuk meminta konfirmasi terkait tindakan yang akan dilakukan.

DUMMY



Gambar 8. Contoh Dialog dengan Pesan dan 2 Tombol
Sumber: developer.android.com

DUMMY

1. **Pentingnya Dialog dalam Aplikasi Android**

   Dialog digunakan untuk memperjelas komunikasi antara aplikasi dan penggunanya. Beberapa hal yang menjadikan dialog sangat berguna adalah:

   a. Pemberitahuan dan Peringatan: Misalnya, ketika pengguna hendak menghapus data yang tidak dapat dipulihkan, dialog dapat menampilkan pesan peringatan yang memastikan bahwa tindakan ini memang disengaja.
   b. Interaksi Langsung: Dialog memungkinkan aplikasi untuk meminta masukan atau keputusan segera dari pengguna, tanpa memindahkan pengguna ke aktivitas baru atau layar yang berbeda.
   c. Peningkatan Pengalaman Pengguna (UX): Pengguna lebih mudah memahami tindakan yang harus diambil ketika aplikasi memberi mereka konteks secara jelas dengan menggunakan dialog.

2. **Jenis-Jenis Dialog dalam Android**

   Android menyediakan beberapa jenis dialog yang dapat digunakan dalam berbagai situasi, bergantung pada kebutuhan aplikasi. Setiap jenis dialog memiliki cara implementasi yang berbeda serta tujuan yang berbeda pula. Berikut ini adalah penjelasan mendalam mengenai jenis-jenis dialog yang dapat digunakan dalam pengembangan aplikasi Android:

   a. *AlertDialog*

      AlertDialog adalah jenis dialog yang paling umum digunakan dalam aplikasi Android. Dialog ini sering digunakan untuk memberi informasi atau meminta konfirmasi dari pengguna terkait tindakan yang akan mereka lakukan. *AlertDialog* biasanya memiliki beberapa tombol pilihan yang memungkinkan pengguna untuk memilih respon yang diinginkan. Dalam implementasinya, *AlertDialog* dapat

menampilkan pesan dengan satu atau dua tombol respons seperti "Ya" dan "Tidak" atau "Setuju" dan "Batal". Hal ini membuat *AlertDialog* sangat ideal digunakan ketika aplikasi perlu memastikan tindakan pengguna sebelum melanjutkan.

Penggunaan *AlertDialog* sangat berguna ketika aplikasi meminta keputusan yang sifatnya penting, seperti mengkonfirmasi penghapusan data atau keluar dari aplikasi. Selain itu, *AlertDialog* juga sering digunakan untuk menampilkan pemberitahuan singkat kepada pengguna.

Contoh Implementasi *AlertDialog*:

```
val Builder = AlertDialog. Builder(this)
 Builder.setTitle("Konfirmasi Penghapusan")
 Builder.setMessage("Apakah Anda yakin ingin menghapus
data ini? Tindakan ini tidak dapat dibatalkan.")
 Builder.setPositiveButton("Ya") { dialog, which ->
 deleteData() // Fungsi untuk menghapus data
}
 Builder.setNegativeButton("Tidak") { dialog, which ->
 dialog.dismiss()  // Menutup dialog jika pengguna
memilih "Tidak"
}
 Builder.show() // Menampilkan dialog
```

Pada contoh di atas, `setTitle` dan `setMessage` digunakan untuk menetapkan judul dan pesan yang akan ditampilkan dalam dialog. Dua tombol, "Ya" dan "Tidak", masing-masing dihubungkan dengan aksi yang sesuai melalui listener pada `setPositiveButton` dan `setNegativeButton`.

b. DatePickerDialog dan TimePickerDialog

Android menyediakan dua jenis dialog untuk memilih tanggal dan waktu, yaitu DatePickerDialog dan TimePickerDialog. Kedua dialog ini sangat berguna dalam aplikasi yang membutuhkan masukan tanggal atau waktu dari pengguna. Sebagai contoh, aplikasi kalender atau aplikasi pemesanan tiket seringkali memerlukan input berupa tanggal dan waktu.

1) DatePickerDialog memungkinkan pengguna memilih tanggal dari antarmuka kalender, yang terdiri dari hari, bulan, dan tahun.
2) TimePickerDialog memungkinkan pengguna memilih waktu dalam format jam dan menit.

Contoh Implementasi DatePickerDialog:

```
val calendar = Calendar.getInstance()
```

DUMMY

```
val year = calendar.get(Calendar.YEAR)
val month = calendar.get(Calendar.MONTH)
val day = calendar.get(Calendar.DAY_OF_MONTH)

val datePickerDialog = DatePickerDialog(this,
 { view, year, monthOfYear, dayOfMonth ->
 val selectedDate = "$dayOfMonth/${monthOfYear + 1}/$year"
 textViewDate.text = "Tanggal Terpilih: $selectedDate"
 }, year, month, day)
datePickerDialog.show()
```

Pada contoh di atas, `Calendar.getInstance()` digunakan untuk mendapatkan tanggal dan waktu saat ini, yang kemudian digunakan sebagai nilai default dalam dialog. Ketika pengguna memilih tanggal, nilai tersebut akan diteruskan dan ditampilkan pada antarmuka aplikasi.

Contoh Implementasi TimePickerDialog:

```
val calendar = Calendar.getInstance()
val hour = calendar.get(Calendar.HOUR_OF_DAY)
val minute = calendar.get(Calendar.MINUTE)

val timePickerDialog = TimePickerDialog(this,
 { view, hourOfDay, minute ->
 val selectedTime = "$hourOfDay:$minute"
 textViewTime.text = "Waktu Terpilih: $selectedTime"
 }, hour, minute, true)
timePickerDialog.show()
```

c. Custom Dialog

Custom Dialog memungkinkan pengembang untuk mendesain dialog dengan tata letak yang disesuaikan (custom layout). Ini sangat berguna ketika tata letak standar seperti yang digunakan oleh *AlertDialog* tidak dapat memenuhi kebutuhan aplikasi. Dengan Custom Dialog, pengembang dapat menentukan elemen UI apa pun yang diinginkan dalam dialog, seperti gambar, form input, atau daftar pilihan.

Dalam implementasi Custom Dialog, kita dapat menggunakan layout XML untuk menentukan struktur tampilan dialog, dan kemudian menghubungkannya dengan kode Java atau Kotlin untuk menangani interaksi pengguna.

Contoh Implementasi Custom Dialog:

```
val dialog = Dialog(this)
```

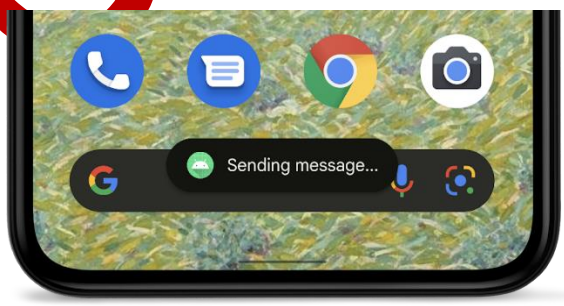
DUMMY

```
dialog.setContentView(R.layout.custom_dialog_layout)
// Mengatur layout dari XML
val buttonOk =
dialog.findViewById<Button>(R.id.buttonOk)
buttonOk.setOnClickListener {
 dialog.dismiss() // Menutup dialog saat tombol
ditekan
}
dialog.show() // Menampilkan dialog
```

B. Toast

Toast adalah elemen antarmuka pengguna dalam aplikasi Android yang digunakan untuk menampilkan pesan singkat kepada pengguna dalam bentuk overlay kecil di layar. Berbeda dengan dialog, toast tidak mengharuskan pengguna untuk mengambil tindakan apapun. Pesan pada toast hanya muncul sementara di layar, umumnya di bagian bawah, dan akan hilang secara otomatis setelah jangka waktu yang singkat. Durasi kemunculan toast dapat diatur secara default menjadi singkat (SHORT) atau panjang (LONG). Toast sangat ideal digunakan untuk memberikan umpan balik atau notifikasi sederhana kepada pengguna tanpa mengganggu alur utama aplikasi.

Toast memainkan peran penting dalam meningkatkan pengalaman pengguna (UX), terutama dalam situasi di mana pengguna perlu diberitahu mengenai suatu peristiwa tanpa harus berhenti atau terganggu oleh notifikasi yang lebih permanen. Pesan yang disampaikan melalui toast biasanya berupa informasi sekilas yang tidak memerlukan perhatian mendalam, seperti notifikasi penyimpanan data atau pesan kesalahan yang sifatnya minor.



Gambar 9. Contoh Toast dengan Pesan 'Sending message…'
(sumber : developer.android.com**)**

1. Karakteristik Toast

DUMMY

a. Sederhana dan Minimalis: Pesan yang muncul di toast bersifat sederhana dan tidak mengganggu aktivitas utama yang sedang berlangsung di layar.
b. Tidak Memerlukan Interaksi Pengguna: Toast muncul sebentar dan menghilang tanpa memerlukan tindakan dari pengguna. Ini menjadikannya ideal untuk pesan yang bersifat sementara dan informatif.
c. Dapat Diatur Desainnya: Android memungkinkan pengembang untuk membuat custom toast, di mana tampilan toast dapat disesuaikan sesuai dengan kebutuhan desain aplikasi.

2. Jenis-Jenis Toast

Ada dua jenis utama toast dalam pengembangan Android, yaitu *default toast* dan *custom toast*. Setiap jenis toast memiliki penggunaan yang berbeda sesuai kebutuhan aplikasi dan pengalaman pengguna yang ingin dicapai.

a. Default Toast

Default Toast adalah jenis toast standar yang ditampilkan di layar menggunakan teks sederhana dan memiliki pengaturan tampilan serta durasi yang ditentukan oleh sistem Android. Dengan menggunakan default toast, pengembang dapat menampilkan pesan sederhana dengan cepat, tanpa harus mengatur tata letak atau gaya khusus. Pesan ini akan muncul di bagian bawah layar selama beberapa detik, kemudian akan menghilang secara otomatis.

Default toast biasanya digunakan untuk pemberitahuan singkat, seperti ketika data berhasil disimpan, atau ketika aplikasi mendeteksi kesalahan yang tidak krusial seperti kegagalan koneksi jaringan sementara.

Contoh Implementasi Default Toast:

```
Toast.makeText(applicationContext, "Data berhasil
disimpan", Toast.LENGTH_SHORT).show()
```

Pada contoh di atas, *makeText* adalah metode bawaan untuk menampilkan toast standar, dengan parameter:

1) *applicationContext* sebagai konteks aplikasi,
2) Pesan teks yang ingin ditampilkan, seperti "Data berhasil disimpan", dan
3) Durasi, yang dapat berupa Toast.LENGTH_SHORT untuk durasi singkat atau Toast.LENGTH_LONG untuk durasi lebih lama.

DUMMY

Penggunaan default toast ini efisien untuk situasi di mana pesan hanya perlu disampaikan secara sekilas tanpa menambahkan elemen visual atau interaksi yang kompleks.

b. Custom Toast

Custom Toast memungkinkan pengembang untuk membuat toast dengan tampilan khusus yang lebih menarik dan informatif. Dengan menggunakan custom toast, pengembang dapat menentukan tampilan toast secara keseluruhan menggunakan layout XML, sehingga dapat menambahkan elemen visual lainnya seperti gambar, warna latar, atau bahkan tata letak yang lebih kompleks sesuai dengan kebutuhan aplikasi. Ini membuat custom toast menjadi pilihan yang lebih baik untuk aplikasi yang membutuhkan keseragaman dalam desain atau ingin memberikan informasi lebih lengkap dibandingkan default toast.

Custom toast berguna dalam konteks tertentu, misalnya, saat aplikasi ingin menarik perhatian pengguna atau menyesuaikan desain UI sesuai dengan tema aplikasi. Dalam custom toast, pengembang dapat memanfaatkan elemen-elemen UI tambahan untuk meningkatkan keterbacaan pesan atau sekadar memperindah tampilannya.

Contoh Implementasi Custom Toast:

```
val layoutInflater = layoutInflater
val layout =
layoutInflater.inflate(R.layout.custom_toast_layout,
findViewById(R.id.custom_toast_container))

val toast = Toast(applicationContext)
toast.duration = Toast.LENGTH_LONG
toast.view = layout
toast.show()
```

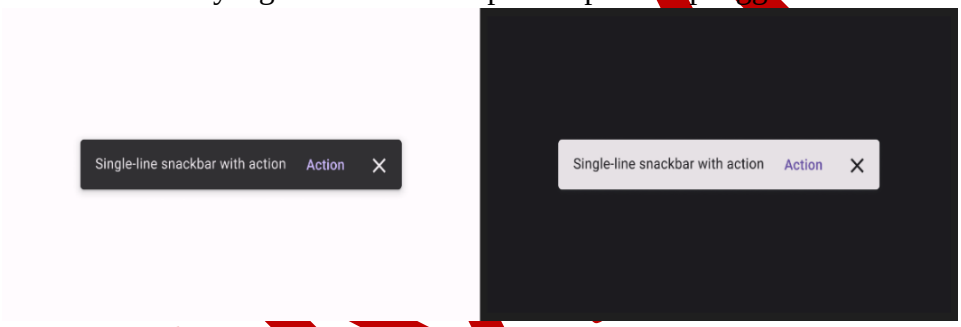
Pada contoh di atas, kita menggunakan metode inflate untuk membuat layout khusus dari file XML custom_toast_layout. Layout ini kemudian diatur sebagai tampilan toast menggunakan toast.view = layout. Dengan menggunakan toast.duration, pengembang juga dapat menentukan lama munculnya toast.

Dalam custom_toast_layout.xml, pengembang dapat mendesain tata letak toast sesuai dengan kebutuhan, misalnya dengan menambahkan teks yang lebih besar, ikon, atau warna latar belakang yang berbeda untuk meningkatkan keterbacaan atau estetika sesuai dengan tema aplikasi.

DRAFT

C. Snackbar

Snackbar adalah salah satu komponen antarmuka pengguna yang digunakan dalam aplikasi Android untuk menampilkan pesan singkat yang memuat notifikasi atau informasi penting, sekaligus menawarkan opsi interaksi bagi pengguna. Secara visual, snackbar muncul di bagian bawah layar dalam bentuk bilah notifikasi yang tidak mengganggu dan akan menghilang setelah beberapa waktu atau jika pengguna mengambil tindakan tertentu. Snackbar mirip dengan toast, tetapi memiliki keunggulan tambahan yaitu memungkinkan pengguna untuk berinteraksi melalui tombol aksi (action button), yang menjadikannya sangat cocok untuk notifikasi yang memerlukan respons cepat dari pengguna.



Gambar 10. Contoh Snackbar
Sumber : developer.android.com

1. Karakteristik Snackbar

Snackbar memiliki beberapa karakteristik yang membedakannya dari toast, antara lain:

a. Kemampuan Interaksi
   Snackbar mendukung tombol aksi yang dapat digunakan pengguna untuk merespons pesan yang diberikan. Misalnya, jika pengguna tidak sengaja menghapus data, snackbar dapat memberikan opsi "Undo" sehingga pengguna dapat membatalkan penghapusan tersebut.
b. Posisi Tampilan yang Konsisten
   Secara default, snackbar muncul di bagian bawah layar dan bertumpang tindih dengan bottom navigation bar atau floating action button jika ada. Posisi ini memberikan pengalaman pengguna yang konsisten dalam menampilkan notifikasi singkat.
c. Durasi Tampilan yang Dapat Diatur

DUMMY

Durasi kemunculan snackbar dapat diatur sesuai kebutuhan, dengan beberapa pilihan durasi standar yang disediakan seperti LENGTH_SHORT, LENGTH_LONG, atau LENGTH_INDEFINITE untuk menampilkan snackbar hingga pengguna menindaklanjutinya atau notifikasi tersebut tidak relevan lagi.

d. Penyesuaian Tampilan

Snackbar dapat disesuaikan tampilannya sesuai dengan tema aplikasi, termasuk warna latar belakang, teks, dan elemen tombol aksi. Hal ini memungkinkan snackbar untuk berbaur dengan keseluruhan antarmuka aplikasi tanpa tampak mencolok atau mengganggu.

2. Jenis-Jenis Snackbar

Snackbar dapat dibagi menjadi dua jenis utama berdasarkan fungsinya, yaitu Snackbar Biasa dan Snackbar dengan Aksi. Setiap jenis memiliki kegunaan yang spesifik, tergantung pada apakah notifikasi yang diberikan membutuhkan respons dari pengguna atau tidak.

a. Snackbar Biasa

Snackbar Biasa adalah snackbar standar yang hanya menampilkan pesan tanpa menyediakan opsi aksi tambahan. Ini digunakan untuk menampilkan notifikasi singkat yang bersifat informatif dan tidak memerlukan tindakan lebih lanjut dari pengguna. Contoh penggunaan snackbar biasa misalnya untuk memberi tahu pengguna bahwa data berhasil dihapus, tetapi pengguna tidak diberikan pilihan untuk mengembalikan data yang dihapus.

Contoh Implementasi Snackbar Biasa :

```
Snackbar.make(findViewById(android.R.id.content),
"Data berhasil dihapus", Snackbar.LENGTH_LONG).show()
```

Dalam kode di atas, Snackbar.make berfungsi untuk membuat snackbar, dengan beberapa parameter:

1) findViewById(android.R.id.content) mengacu pada tampilan yang menjadi dasar dari snackbar.
2) Pesan teks yang akan ditampilkan, dalam hal ini "Data berhasil dihapus".
3) Durasi tampilannya, misalnya Snackbar.LENGTH_LONG untuk tampilan yang agak lama.

DUMMY

Snackbar biasa umumnya digunakan untuk menyampaikan informasi sekilas yang tidak memerlukan tindak lanjut dari pengguna, seperti pemberitahuan kesuksesan operasi atau penyelesaian tindakan tertentu.

b. Action Snackbar

Snackbar dengan Aksi memiliki tombol tambahan yang memungkinkan pengguna untuk merespons pesan yang ditampilkan. Tombol ini sering digunakan untuk tindakan seperti "Undo" (batalkan) atau "Retry" (ulang), yang memberikan pengguna kesempatan untuk memperbaiki tindakan atau mengulangi operasi yang telah dilakukan. *Action Snackbar* sangat cocok digunakan dalam situasi di mana pengguna mungkin perlu membatalkan atau mengulang tindakan tertentu, seperti penghapusan data atau pengiriman pesan.

Contoh Implementasi Action Snackbar:

```
Snackbar.make(findViewById(android.R.id.content),
"Data berhasil dihapus", Snackbar.LENGTH_LONG)
.setAction("Undo") {
 // Aksi yang dilakukan ketika pengguna menekan tombol
"Undo"
}.show()
```

Pada contoh ini, metode setAction digunakan untuk menambahkan tombol aksi dengan teks "Undo". Jika pengguna menekan tombol tersebut, kode yang ada di dalam blok { ... } akan dijalankan. Pada situasi ini, aksi yang dapat dilakukan misalnya mengembalikan data yang dihapus.

Penggunaan action snackbar memberikan pengguna kontrol lebih besar dalam pengalaman aplikasi, karena pengguna dapat secara langsung memperbaiki kesalahan atau mengambil keputusan sesuai konteks.

D. ActionBar

ActionBar adalah komponen antarmuka pengguna yang berada di bagian atas layar dan berfungsi sebagai bar navigasi serta kontrol utama aplikasi. Biasanya, ActionBar menampilkan judul aplikasi, ikon navigasi, dan menu tindakan, sehingga memudahkan pengguna dalam mengakses fitur-fitur inti aplikasi. Selain itu,

1. Jenis-jenis Navigasi dalam ActionBar

DUMMY

Android menyediakan beberapa jenis navigasi dalam ActionBar yang sesuai dengan berbagai kebutuhan aplikasi:

a. Navigasi Up: Tombol yang memungkinkan pengguna untuk kembali ke layar sebelumnya.
b. Navigasi Tab: Berfungsi untuk mengelompokkan konten atau tampilan dalam bentuk tab, memungkinkan navigasi antar-tab dengan mudah.
c. Drop Down List: Memungkinkan pengguna untuk memilih item dari daftar dropdown langsung di ActionBar.

2. Contoh Implementasi ActionBar

Untuk menggunakan ActionBar dalam aplikasi Android, kita dapat mengimplementasikan menu dengan menambahkan item ke dalam ActionBar melalui metode onCreateOptionsMenu() dan menentukan aksi menu dalam metode onOptionsItemSelected().

a. Menambahkan Item Menu

Metode onCreateOptionsMenu() digunakan untuk menambahkan item ke menu ActionBar dengan cara memuat (inflate) *resource* menu dari file XML.Ketik konten Sub Bab disini.

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean
{
 menuInflater.inflate(R.menu.main_menu, menu)
 return true
}
```

Pada contoh ini, menuInflater.inflate(R.menu.main_menu, menu) berfungsi untuk membuat file XML main_menu.xml sebagai menu yang akan ditampilkan di ActionBar.

b. Menambahkan Item Menu

Metode onOptionsItemSelected() digunakan untuk mendefinisikan tindakan yang akan dilakukan ketika suatu item menu dipilih.

```
override fun onOptionsItemSelected(item: MenuItem):
Boolean {
 return when (item.itemId) {
 R.id.action_settings -> {
  // Aksi yang dijalankan saat menu "Settings" dipilih
  true
 }
 else -> super.onOptionsItemSelected(item)
 }
}
```

DUMMY

Pada contoh ini, when (item.itemId) memeriksa ID dari item menu yang dipilih, dan jika R.id.action_settings dipilih, maka aksi tertentu akan dijalankan.

3. **Penyesuaian Tampilan ActionBar**

   Android memungkinkan beberapa penyesuaian tampilan untuk ActionBar agar sesuai dengan kebutuhan aplikasi:

   a. Mengganti Judul: Judul ActionBar dapat diubah dengan menggunakan metode setTitle().

   ```
   supportActionBar?.title = "Halaman Utama"
   ```

   b. Mengubah Warna Latar dan Teks: Pengembang dapat mengubah warna latar belakang ActionBar agar sesuai dengan tema aplikasi melalui *resource* styles.xml.

   c. Menambah Ikon Kustom: Ikon pada ActionBar dapat disesuaikan atau ditambahkan menggunakan metode setDisplayHomeAsUpEnabled() untuk menampilkan ikon panah balik

E. Multithreading

Multithreading adalah teknik pemrograman yang memungkinkan suatu aplikasi untuk menjalankan beberapa tugas secara paralel atau bersamaan. Dalam konteks aplikasi Android, teknik ini sangat penting untuk menjaga performa dan responsivitas aplikasi, terutama ketika aplikasi harus menangani tugas-tugas berat seperti pengolahan data dalam jumlah besar, pengambilan data dari jaringan, atau tugas-tugas lain yang memerlukan waktu eksekusi panjang. Tanpa multithreading, semua tugas akan berjalan di main thread atau UI thread, yang berfokus pada pemrosesan antarmuka pengguna. Hal ini dapat mengakibatkan aplikasi menjadi tidak responsif atau bahkan memicu error "Application Not Responding" (ANR) jika main thread terlalu lama menangani tugas berat.

1. **Pentingnya Multithreading**

   Android memiliki sistem operasi yang memprioritaskan responsivitas aplikasi. Main thread atau UI thread bertanggung jawab untuk menangani semua tugas yang berkaitan dengan antarmuka pengguna (UI), termasuk rendering komponen visual, respons terhadap interaksi pengguna, dan pembaruan tampilan. Jika ada tugas berat yang dijalankan di main thread, seperti pengambilan data dari API atau

DUMMY

pengolahan file besar, thread ini akan terbebani dan mengakibatkan UI menjadi lambat atau bahkan tidak merespons sama sekali. Dengan menggunakan multithreading, tugas-tugas tersebut dapat dijalankan di background thread, sehingga UI tetap berjalan dengan lancar dan responsif.

2. Jenis-Jenis Multithreading

Multithreading di Android dapat dilakukan dengan beberapa cara. Berikut ini adalah metode umum yang sering digunakan dalam pengembangan aplikasi Android:

a. Thread dan Handler

1) Thread adalah kelas dasar di Java yang digunakan untuk menjalankan tugas-tugas secara paralel. Pada implementasi dasar, Anda bisa membuat instance dari kelas Thread dan meng-override metode run(), yang akan menjalankan kode secara paralel di background thread.
2) Handler berfungsi untuk mengirim pesan atau menjalankan kode pada thread yang ditargetkan. Dalam kasus ini, Handler sering digunakan untuk mengembalikan kontrol ke main thread setelah tugas di background thread selesai, sehingga pembaruan UI dapat dilakukan.

   Contoh sederhana implementasi menggunakan Thread dan Handler:

```
val handler = Handler(Looper.getMainLooper())

Thread {
 // Tugas berat dijalankan di background thread
 val result = performHeavyTask()

 // Kembali ke main thread untuk memperbarui UI
 handler.post {
 updateUI(result)
 }
}.start()
```

b. AsyncTask (Deprecated)

AsyncTask sebelumnya populer untuk menangani tugas-tugas ringan di background dan kembali ke main thread setelah selesai. AsyncTask menyediakan metode untuk tugas-tugas yang dilakukan di background (doInBackground) dan pembaruan UI di main thread (onPostExecute).

DRAFT

Namun, AsyncTask mulai ditinggalkan karena keterbatasannya dalam pengelolaan siklus hidup dan rentan terhadap masalah memori. Tugas yang dijalankan melalui AsyncTask dapat tetap aktif walaupun *Activity* atau *Fragment* terkait dihancurkan, sehingga berpotensi menyebabkan memory leaks. Dengan alasan ini, AsyncTask tidak lagi direkomendasikan, dan banyak pengembang beralih ke Kotlin Coroutine sebagai penggantinya.
Berikut adalah contoh penggunaan AsyncTask:

```
class MyAsyncTask : AsyncTask<Void, Void, String>() {
 override fun doInBackground(vararg params: Void?):
String {
 // Tugas berat
 return performHeavyTask()
 }

 override fun onPostExecute(result: String) {
 // Pembaruan UI di main thread
 updateUI(result)
 }
}
```

c. Kotlin Coroutine
Kotlin Coroutine adalah pendekatan modern dan lebih efisien untuk mengelola tugas-tugas yang berjalan secara paralel di Android. Coroutines menawarkan sintaks yang lebih bersih dan penanganan memori yang lebih baik, serta dirancang dengan mempertimbangkan lifecycle-aware, yang berarti mereka dapat menghentikan tugas ketika *Activity* atau *Fragment* dihancurkan.
Coroutine sangat fleksibel dan menyediakan dua dispatcher utama: Dispatchers.IO untuk tugas-tugas yang memerlukan input/output (misalnya, pengambilan data dari internet atau akses database), dan Dispatchers.Main untuk tugas-tugas yang berkaitan dengan UI.
Berikut adalah contoh sederhana penggunaan Coroutine:

```
CoroutineScope(Dispatchers.IO).launch {
 // Jalankan tugas berat di background thread
 val result = performHeavyTask()

 withContext(Dispatchers.Main) {
 // Perbarui antarmuka pengguna di main thread
 updateUI(result)
 }
}
```

Contoh diatas menampilkan CoroutineScope (Dispatchers.IO).launch, yang menjalankan tugas berat di Dispatchers.IO. Fungsi withContext(Dispatchers.Main) digunakan untuk kembali ke main thread setelah tugas berat selesai, sehingga pembaruan UI dapat dilakukan.

Coroutine di Kotlin berfungsi sebagai "lightweight thread" yang memungkinkan penulisan kode asinkron dengan lebih sederhana, menghilangkan kebutuhan callback yang berbelit-belit, seperti yang sering terjadi pada Thread atau AsyncTask. Keunggulan dari Coroutine adalah kemampuannya untuk "suspend" atau menunda eksekusi sementara, yang memungkinkan pengelolaan tugas paralel dengan efisien tanpa membebani *resource* perangkat.

Berikut ini adalah contoh lengkap penggunaan Coroutine dalam skenario pengambilan data dari jaringan:

```
fun fetchData() {
 CoroutineScope(Dispatchers.IO).launch {
 // Mengambil data dari jaringan
 val data = getDataFromNetwork()

 // Kembali ke main thread untuk memperbarui UI
 withContext(Dispatchers.Main) {
  displayData(data)
 }
 }
}

suspend fun getDataFromNetwork(): String {
 // Simulasi operasi jaringan
 delay(2000)
 return "Data dari jaringan"
}

fun displayData(data: String) {
 // Update UI dengan data
 textView.text = data
}
```

Pada contoh ini, fungsi fetchData() menggunakan getDataFromNetwork() di background menggunakan Dispatchers.IO, lalu mengembalikan hasil ke main thread menggunakan withContext(Dispatchers.Main) untuk memperbarui tampilan UI dengan data yang diperoleh.

F. Implementasi Dialog, Toast, Snackbar, Actionbar, dan Multhithreading

Langkah-Langkah Implementasi Aplikasi

1. **Persiapan Proyek Android Studio**
   Langkah 1.1: Buka Android Studio
   Langkah 1.2: Beri nama package topik4 dengan cara klik kanan pada package **com.kurniawan.belajarandroidkotlin** lalu klik **New > package >**
2. **Memperbarui code pada Main *Activity*.kt**
   Langkah 2.1: Buka file **Main *Activity*.kt** yang secara otomatis sudah ada di dalam folder **kotlin+java/com.kurniawan.belajarandroidkotlin.**
   Langkah 2.2: Tambahkan kode berikut untuk mendefinisikan tombol navigasi *Activity* DialogKetik konten Sub Bab disini

MainActivity.kt

```
package com.kurniawan.belajarandroidkotlin

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.FragmentActivity
import com.kurniawan.belajarandroidkotlin.topik4.DialogActivity
import com.kurniawan.belajarandroidkotlin.topik5.RecycleActivity

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val buttonStart = findViewById<Button>(R.id.button_start)
        buttonStart.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, SecondActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan3 = findViewById<Button>(R.id.button_pertemuan3)
        buttonPertemuan3.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, IntentActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan4 = findViewById<Button>(R.id.button_pertemuan4)
        buttonPertemuan4.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, FragmentActivity::class.java)
            startActivity(intent)
        }
        val buttonDialog = findViewById<Button>(R.id.buttonDialog)
        buttonDialog.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, DialogActivity::class.java)
            startActivity(intent)
        }
        val buttonAdpater = findViewById<Button>(R.id.buttonAdapter)
        buttonAdpater.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, RecycleActivity::class.java)
            startActivity(intent)
        }
    }
}
```

3. **Mengatur Layout *Activity*_main.xml**

Langkah 3.1: Buka file ***Activity*_main.xml** di dalam folder **res/layout/.**

Langkah 3.2: Tambahkan kode XML berikut untuk membuat tampilan Dialog button pada layout.

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

 <TextView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Belajar Basic Android dengan Kotlin"
 android:layout_marginTop="10dp"
 android:layout_marginLeft="10dp"
 android:textSize="15dp"
 android:gravity="center" />

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp">
 <Button
  android:id="@+id/button_start"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Second Activity" />
 </LinearLayout>

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="10dp">
 <Button
  android:id="@+id/button_pertemuan3"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="Intent Activity" />
 </LinearLayout>

 <LinearLayout
 android:layout_width="match_parent"
```

```
    android:layout_height="wrap_content"
    android:layout_margin="10dp">
    <Button
     android:id="@+id/button_pertemuan4"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Fragment Activity" />
    </LinearLayout>

    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp">
    <Button
     android:id="@+id/buttonDialog"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Dialog Activity" />
    </LinearLayout>

    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp">
    <Button
     android:id="@+id/buttonAdapter"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:text="Go To Adapter Content" />
    </LinearLayout>

   </LinearLayout>
```

4. **Implementasi Dialog**

   Langkah 4.1: Buat file **Dialog *Activity*.kt** di dalam package topik4 dengan cara klik kanan pada package, pilih **New > *Activity* > Empty Views *Activity*.**

   Langkah 4.2: Pada jendela pengaturan ***Activity***, lakukan konfigurasi berikut:

   - Pada opsi ***Activity* Name**, ketik **Dialog *Activity*.**
   - Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
   - Pada bagian **Layout Name**, ketik ***Activity*_dialog.xml** sebagai nama layout.
   - Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.

- Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.topik4**
- Pada bagian Source Language, pastikan bahasa yang dipilih adalah Kotlin.

Langkah 4.3: Setelah semua pengaturan sesuai, klik **Finish** untuk membuat *Activity* baru beserta layout-nya.

Langkah 4.4: Tambahkan kode berikut pada Dialog *Activity*.kt:

DialogActivity.kt

```
package com.kurniawan.belajarandroidkotlin.topik4

import android.content.Intent
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import android.widget.Button
import android.widget.Toast
import androidx.activity.addCallback
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.snackbar.Snackbar
import com.kurniawan.belajarandroidkotlin.R

class DialogActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dialog)

        supportActionBar?.title = "BelajarAndroidKotlin"
        supportActionBar?.setDisplayHomeAsUpEnabled(true)

        onBackPressedDispatcher.addCallback( owner: this) { this: OnBackPressedCallback
            finish()
        }

        val toastButton = findViewById<Button>(R.id.btnToast)
        val snackbarButton = findViewById<Button>(R.id.btnSnackbar)
        val dialogButton = findViewById<Button>(R.id.btnDialog)

        toastButton.setOnClickListener { it: View!
            Toast.makeText( context: this,  text: "Ini adalah Toast", Toast.LENGTH_SHORT).show()
        }

        snackbarButton.setOnClickListener { it: View!
            Snackbar.make(it,  text: "Ini adalah Snackbar", Snackbar.LENGTH_LONG)
                .setAction( text: "Tutup") { it: View!
                    Toast.makeText( context: this,  text: "Snackbar Ditutup", Toast.LENGTH_SHORT).show()
                }.show()
        }

        dialogButton.setOnClickListener { it: View!
            val builder = AlertDialog.Builder( context: this)
            builder.setTitle("Dialog Contoh")
```

DialogActivity.kt

```
                .setAction( text: "Tutup") { it: View!
                    Toast.makeText( context: this,  text: "Snackbar Ditutup", Toast.LENGTH_SHORT).show(
                }.show()
            }

            dialogButton.setOnClickListener { it: View!
                val builder = AlertDialog.Builder( context: this)
                builder.setTitle("Dialog Contoh")
                builder.setMessage("Apakah Anda ingin melanjutkan?")
                builder.setPositiveButton( text: "Ya") { _, _ ->
                    Toast.makeText( context: this,  text: "Melanjutkan", Toast.LENGTH_SHORT).show()
                }
                builder.setNegativeButton( text: "Tidak") { _, _ ->
                    Toast.makeText( context: this,  text: "Dibatalkan", Toast.LENGTH_SHORT).show()
                }
                builder.show()
            }
        }

        override fun onCreateOptionsMenu(menu: Menu?): Boolean {
            menuInflater.inflate(R.menu.menu_dialog_activity, menu)
            return true
        }

        override fun onOptionsItemSelected(item: MenuItem): Boolean {
            return when (item.itemId) {
                R.id.action_settings -> {
                    Toast.makeText( context: this,  text: "Pengaturan dipilih", Toast.LENGTH_SHORT).show()
                    true
                }
                R.id.action_help -> {
                    val intent = Intent( packageContext: this, HelpActivity::class.java)
                    startActivity(intent)
                    true
                }
                else -> super.onOptionsItemSelected(item)
            }
        }
    }

```

Langkah 4.5: Buat layout *Activity*_dialog.xml di folder res/layout/ untuk Dialog *Activity*:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/andro
id"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".topik4.Dialog Activity">

 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 android:gravity="center">
```

```
    <Button
    android:id="@+id/btnToast"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Toast" />

    <Button
    android:id="@+id/btnSnackbar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Snackbar" />

    <Button
    android:id="@+id/btnDialog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Dialog" />
    </LinearLayout>
</RelativeLayout>
```

5. **Implementasi Help Setting**

Langkah 5.1: Buat file **Help *Activity*.kt** di dalam package topik4 dengan cara klik kanan pada package, pilih **New > *Activity* > Empty Views *Activity***.

Langkah 5.2: Pada jendela pengaturan ***Activity*,** lakukan konfigurasi berikut:

- Pada opsi ***Activity* Name**, ketik **Help *Activity***.
- Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
- Pada bagian **Layout Name**, ketik ***Activity*_help.xml** sebagai nama layout.
- Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.
- Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.topik4**
- Pada bagian **Source Language**, pastikan bahasa yang dipilih adalah **Kotlin.**

Langkah 5.3: Setelah semua pengaturan sesuai, klik **Finish** untuk membuat *Activity* baru beserta layout-nya.

Langkah 5.4: Tambahkan kode berikut pada **Help *Activity*.kt:**

DUMMY

HelpActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik4

import android.os.Bundle
import androidx.activity.addCallback
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R

class HelpActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_help)

        supportActionBar?.title = "Halaman Bantuan"
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        onBackPressedDispatcher.addCallback( owner: this) { this: OnBackPressedCallback
            finish()
        }
    }
    override fun onSupportNavigateUp(): Boolean {
        finish()
        return true
    }
}

```

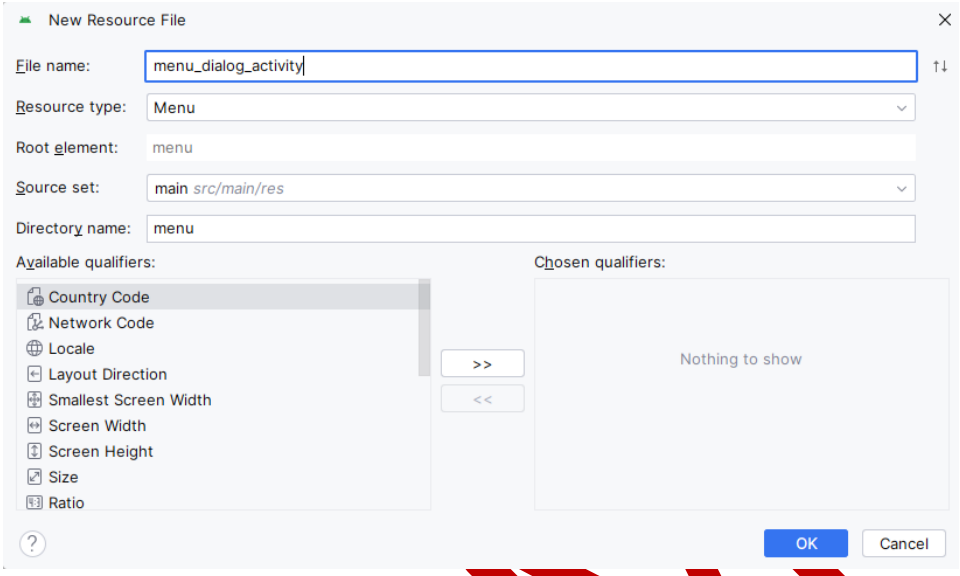
Langkah 5.5: Buat layout ***Activity*_help.xml** di folder res/layout/ untuk Help *Activity*:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".topik4.Help Activity">

    <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Ini adalah halaman bantuan."
    android:textSize="18sp" />
</LinearLayout>
```

6. Implementasi Action Bar

Langkah 6.1: Buat package baru pada bagian res dengan cara klik kanan , pilih **New > Android *resource* File**

Langkah 6.2: Tambahkan *resource* File seperti berikut lalu klik OK:



7. **Mengatur Layout menu_dialog_ *Activity*.xml**

   Langkah 7.1: Buka **menu_dialog_ *Activity*.xml** di dalam folder **res/menu/.**

   Langkah 7.2: Tambahkan kode XML berikut untuk mengatur tampilan pada Action Bar:

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/andro
id">
 <item
  android:id="@+id/action_settings"
  android:title="Pengaturan"
  android:orderInCategory="100"
  android:showAsAction="never" />
  <item
  android:id="@+id/action_help"
  android:title="Bantuan"
  android:orderInCategory="100"
  android:showAsAction="never"/>
</menu>
```

8. **Mengkonfigurasi AndroidManifest.xml**

   Langkah 8.1: Tambahkan deklarasi setaip *Activity* di **AndroidManifest.xml** untuk mengenali file-file ini dalam aplikasi:

```
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:label="BelajarAndroidKotlin"
            android:theme="@style/Theme.AppCompat.Light.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".topik4.HelpActivity"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.Light" />
        <activity
            android:name=".topik4.DialogActivity"
            android:exported="true"
            android:theme="@style/Theme.AppCompat.Light" />
        <activity
            android:name=".FragmentActivity"
            android:exported="false" />
        <activity
            android:name=".ExplicitActivity"
            android:exported="true" />
        <activity
            android:name=".ImplicitActivity"
            android:exported="true" />
        <activity
            android:name="com.kurniawan.belajarkotlin.ResultExplicitActivity"
            android:exported="false" />
        <activity
            android:name=".IntentActivity"
            android:exported="true" />
        <activity
            android:name=".SecondActivity"
            android:exported="false" />
    </application>
```

9. Menjalankan Dan Menguji Aplikasi

Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.

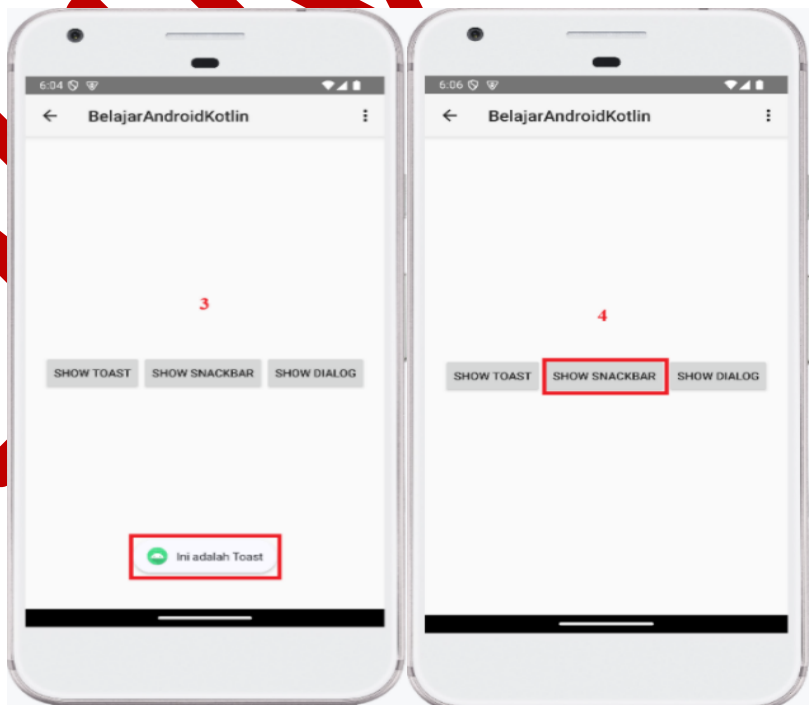
Langkah 9.1: Klik Run pada toolbar Android Studio atau tekan Shift + F10 untuk memulai proses kompilasi dan menjalankan aplikasi.

Langkah 9.2: Pada jendela Select Deployment Target, pilih perangkat yang akan digunakan untuk menguji aplikasi:

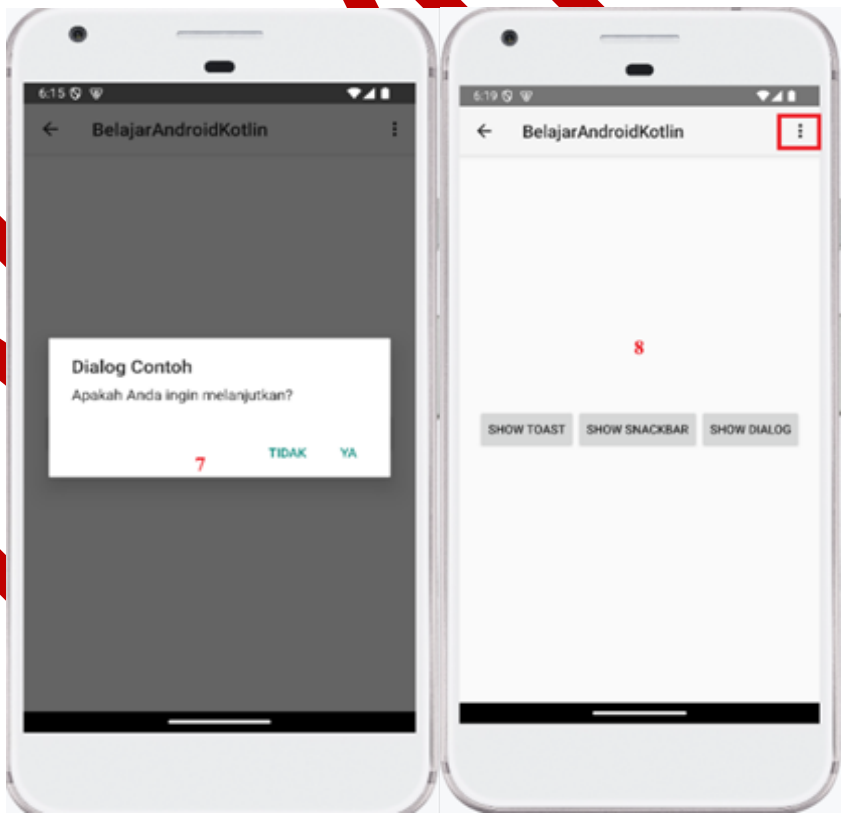
a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode USB Debugging aktif.

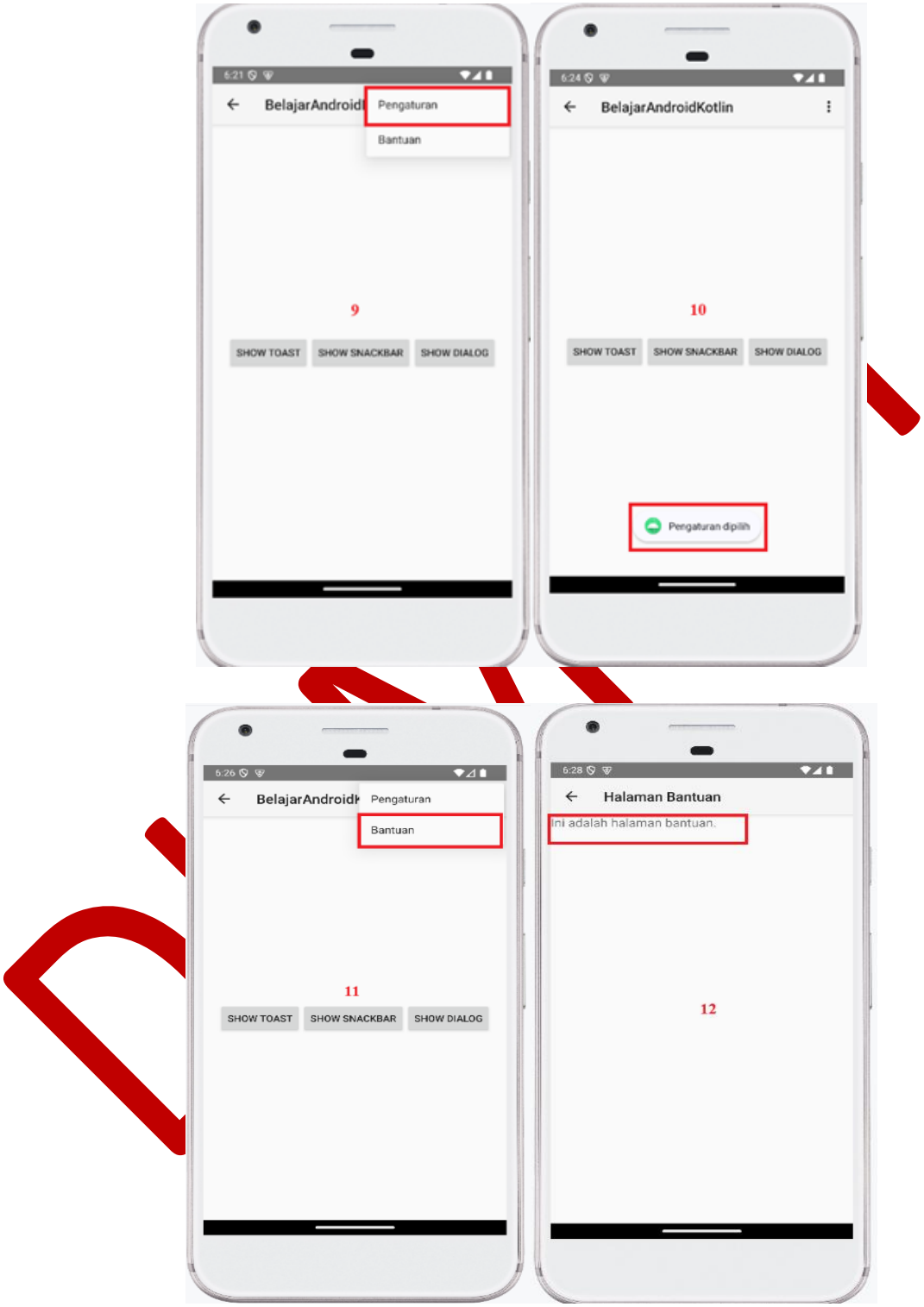
10. Hasil Pengujian

Setelah aplikasi berhasil dijalankan, Anda akan melihat tampilan berikut pada layar perangkat atau emulator:




BelajarAndroidKotlin
5
SHOW TOAST
SHOW SNACKBAR
SHOW DIALOG
Ini adalah Snackbar
TUTUP
BelajarAndroidKotlin
6
SHOW TOAST
SHOW SNACKBAR
SHOW DIALOG


BelajarAndroidKotlin
Dialog Contoh
Apakah Anda ingin melanjutkan?
TIDAK
YA
7
BelajarAndroidKotlin
8
SHOW TOAST
SHOW SNACKBAR
SHOW DIALOG



Gambar 11. Dialog, Toast, Snacbar, Actionbar dan Multithreading

G. Tugas

1. **Modifikasi** : Setelah menyelesaikan setiap dialog di atas lanjutkan pembuatan toast yang mampu menampilkan notifikasi dengan latar warna yang berbeda
   Contoh : Toast error berlatar merah untuk menampilkan pesan error
   Toast success berlatar hijau untuk menampilkan pesan sukses
2. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 7
ADAPTER DAN *RECYCLERVIEW*

Adapter adalah komponen yang menghubungkan data dengan tampilan dalam *RecyclerView*. *RecyclerView* sendiri adalah widget tampilan di Android yang digunakan untuk menampilkan daftar data secara efisien dan dinamis. Berbeda dengan komponen sebelumnya seperti ListView, *RecyclerView* mendukung berbagai jenis layout seperti LinearLayout, GridLayout, dan StaggeredGridLayout, serta memungkinkan penggunaan ViewHolder untuk mendaur ulang tampilan item yang tidak terlihat di layar, mengurangi penggunaan memori dan meningkatkan performa. *Adapter* bertugas untuk mengelola data, menghubungkan data dengan tampilan setiap item dalam daftar, dan menangani interaksi pengguna seperti klik pada item. Dengan kombinasi *Adapter* dan *RecyclerView*, pengembang dapat menciptakan daftar yang interaktif, fleksibel, dan efisien dalam aplikasi Android.

Selain fungsinya yang utama sebagai penghubung antara data dan tampilan, *Adapter* juga memungkinkan pengembang untuk menyesuaikan tampilan item dalam daftar sesuai dengan kebutuhan aplikasi. Setiap item dalam *RecyclerView* dapat dikustomisasi melalui layout yang disebut item layout, yang bisa berisi berbagai elemen UI seperti teks, gambar, atau tombol. *Adapter* akan mengikat data dengan elemen-elemen ini, sehingga setiap item dalam daftar bisa menampilkan informasi yang relevan dan dinamis. *RecyclerView* juga menyediakan berbagai fitur tambahan seperti animasi item saat menambah atau menghapus data, serta dukungan untuk fitur interaktif seperti drag and drop atau swipe untuk menghapus item. Semua fitur ini menjadikan *RecyclerView* dan *Adapter* kombinasi yang sangat kuat dalam membangun antarmuka pengguna yang efisien dan responsif di aplikasi Android.

A. *Adapter*

Adapter adalah komponen yang digunakan untuk menghubungkan data dengan tampilan dalam *RecyclerView*. *Adapter* bertugas untuk menyediakan tampilan untuk setiap item dalam daftar dan menghubungkannya dengan data yang sesuai. *Adapter* bertanggung jawab untuk mengelola dan menampilkan data secara dinamis di setiap elemen tampilan (item) di dalam *RecyclerView*. Tanpa *Adapter*, *RecyclerView* tidak dapat menampilkan data yang dimilikinya, karena *Adapter* yang

DUMMY

akan mengatur bagaimana data tersebut ditampilkan di setiap item dalam daftar.

Fungsi *Adapter*:

1. Mengelola Data
   Adapter bertanggung jawab untuk menyiapkan dan menyediakan data yang akan ditampilkan dalam *RecyclerView*.
2. Membuat Tampilan untuk Setiap Item
   Adapter membuat tampilan untuk setiap item yang ada dalam daftar, berdasarkan layout item yang telah ditentukan.
3. Menghubungkan Data ke Tampilan
   Adapter mengikat data yang ada dengan tampilan item di *RecyclerView*, misalnya dengan menampilkan nama, gambar, atau informasi lainnya yang sesuai dengan posisi item dalam daftar.
4. Menangani Interaksi Pengguna
   Adapter dapat menangani interaksi pengguna, seperti klik pada item, untuk mengeksekusi perintah tertentu berdasarkan data yang terkait dengan item tersebut.

Struktur *Adapter*:

1. **onCreateViewHolder()**: Digunakan untuk membuat tampilan baru dari layout item.
2. **onBindViewHolder():** Digunakan untuk mengikat data ke tampilan item yang sudah ada.
3. **getItemCount():** Menyediakan jumlah item yang ada di dalam data.

B. *RecyclerView*

RecyclerView adalah widget tampilan di Android yang digunakan untuk menampilkan daftar data dalam bentuk yang dapat digulir (scrollable). *RecyclerView* merupakan pengembangan dari ListView dan GridView, namun menawarkan fleksibilitas yang lebih tinggi serta performa yang lebih efisien. *RecyclerView* memungkinkan penataan data dalam berbagai layout, seperti LinearLayout (daftar vertikal atau horizontal), GridLayout (daftar berbentuk grid), atau StaggeredGridLayout (daftar dengan item yang tidak sejajar secara vertikal atau horizontal).

Keunggulan utama *RecyclerView* dibandingkan dengan ListView adalah kemampuannya untuk mendaur ulang tampilan item yang sudah tidak terlihat di layar, menggunakan konsep ViewHolder. Hal ini sangat

DUMMY

mengurangi penggunaan memori dan meningkatkan performa aplikasi, terutama ketika menampilkan daftar data dalam jumlah besar.

Komponen Utama dalam *RecyclerView*:

1. ***RecyclerView*:** Komponen utama yang menampilkan daftar data. *RecyclerView* menggantikan ListView dan GridView dengan performa yang lebih baik.
2. **LayoutManager:** Mengatur bagaimana item ditampilkan di *RecyclerView*. Beberapa jenis LayoutManager yang sering digunakan:
3. **LinearLayoutManager:** Menampilkan item secara vertikal atau horizontal.
4. **GridLayoutManager:** Menampilkan item dalam bentuk grid (kolom dan baris).
5. **StaggeredGridLayoutManager:** Menampilkan item dalam bentuk grid dengan ukuran yang tidak seragam (item bisa lebih besar atau lebih kecil).
6. ***Adapter*:** Menghubungkan data dengan tampilan di *RecyclerView*. *Adapter* akan menyediakan data dan tampilan untuk setiap item.
7. **ViewHolder:** Merupakan objek yang menyimpan referensi ke tampilan item di *RecyclerView*. Dengan menggunakan ViewHolder, *RecyclerView* dapat mendaur ulang tampilan item yang sudah tidak terlihat, sehingga meningkatkan efisiensi memori.

C. Implementasi *Adapter* dan *RecyclerView*

Langkah-Langkah Implementasi Aplikasi

1. **Persiapan Proyek Android Studio**

   Langkah 1.1: Buka project **BelajarAndroidKotlin** yang sudah dibuat sebelumnya.

   Langkah 1.2: Beri nama package **topik5** dengan cara klik kanan pada package **com.kurniawan.belajarandroidkotlin** lalu klik **New > package**
2. **Membuat File-File Utama**

   Langkah 2.1: Buka file **Main *Activity*.kt** yang sudah dibuat sebelumnya.

   Langkah 2.2: Tambahkan kode berikut untuk mendefinisikan tombol navigasi *Activity Adapter*

DUMMY

```
package com.kurniawan.belajarandroidkotlin

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.FragmentActivity
import com.kurniawan.belajarandroidkotlin.topik4.DialogActivity
import com.kurniawan.belajarandroidkotlin.topik5.RecycleActivity

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val buttonStart = findViewById<Button>(R.id.button_start)
        buttonStart.setOnClickListener {
            val intent = Intent( this, SecondActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan3 = findViewById<Button>(R.id.button_pertemuan3)
        buttonPertemuan3.setOnClickListener {
            val intent = Intent( this, IntentActivity::class.java)
            startActivity(intent)
        }

        val buttonPertemuan4 = findViewById<Button>(R.id.button_pertemuan4)
        buttonPertemuan4.setOnClickListener {
            val intent = Intent( this, FragmentActivity::class.java)
            startActivity(intent)
        }
        val buttonDialog = findViewById<Button>(R.id.buttonDialog)
        buttonDialog.setOnClickListener {
            val intent = Intent( this, DialogActivity::class.java)
            startActivity(intent)
        }
        val buttonAdpater = findViewById<Button>(R.id.buttonAdapter)
        buttonAdpater.setOnClickListener {
            val intent = Intent( this, RecycleActivity::class.java)
            startActivity(intent)
        }
    }
}
```

Langkah 2.3: Buat layout ***Activity*_main.xml** di folder **res/layout/** untuk navigasi Main *Activity* tombol *Adapter*:

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

 <TextView
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Belajar Basic Android dengan Kotlin"
android:layout_marginTop="10dp"
android:layout_marginLeft="10dp"
android:textSize="15dp"
android:gravity="center" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp">
<Button
 android:id="@+id/button_start"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Second Activity" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp">
<Button
 android:id="@+id/button_pertemuan3"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Intent Activity" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp">
<Button
 android:id="@+id/button_pertemuan4"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Fragment Activity" />
</LinearLayout>
<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp">
<Button
 android:id="@+id/buttonDialog"
 android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
    android:text="Dialog Activity" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp">
<Button
    android:id="@+id/buttonAdapter"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Adapter Dan RecyclerView Activity" />
</LinearLayout>
</LinearLayout>
```

3. **Implementasi *RecyclerView* Dan *Adapter***

Langkah 3.1: Buat file **Recycle *Activity*.kt** di dalam package topik5 dengan cara klik kanan pada package, pilih **New** > ***Activity*** > **Empty Views *Activity*.**

Langkah 3.2: Pada jendela pengaturan ***Activity*,** lakukan konfigurasi berikut:

a. Pada opsi ***Activity* Name**, ketik **Recycle *Activity*.**
b. Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
c. Pada bagian **Layout Name**, ketik ***Activity*_recycle.xml** sebagai nama layout.
d. Abaikan bagian Launcher *Activity* karena tidak perlu diubah.
e. Pastikan **Package Name** berada pada package com.kurniawan.belajarandroid kotlin.topik5
f. Pada bagian **Source Language**, pastikan bahasa yang dipilih adalah **Kotlin**.

Langkah 3.3: Setelah semua pengaturan sesuai, klik Finish untuk membuat *Activity* baru beserta layout-nya.

Langkah 3.4: Tambahkan kode berikut pada **Recycle *Activity*.kt:**

DUMMY

RecycleActivity.kt

```
package com.kurniawan.belajarandroidkotlin.topik5

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kurniawan.belajarandroidkotlin.R

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

        val fruitList = listOf(
            User( name: "Ade Kurniawan", R.drawable.ade_kurniawan),
            User( name: "Jusmardi", R.drawable.jusmardi),
            User( name: "Novi Febrianti", R.drawable.novi_febrianti),
            User( name: "Yeka Hendriyani", R.drawable.yeka_hendriyani)
        )

        val recyclerView = findViewById<RecyclerView>(R.id.rvUser)
        recyclerView.layoutManager = LinearLayoutManager( context: this)
        recyclerView.adapter = FruitAdapter(fruitList)
    }
}
```

Langkah 3.5: Buat layout *Activity*_recycle.xml di folder res/layout/ untuk Recycle *Activity*:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="vertical"
 android:padding="16dp">
 <androidx.RecyclerView.widget.RecyclerView
 android:id="@+id/rvUser"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp" />
</LinearLayout>
```

4. **Membuat Model Dan *Adapter***

Langkah 4.1: Buat data class **User.kt** di dalam package *topik5* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File**

Langkah 4.2: Tambahkan kode berikut pada **User.kt:**

User.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik5

data class User(
    val name: String,
    val image: Int
)
```

Langkah 4.3: Buat data class **User*Adapter*.kt** di dalam package *topik5* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File**

Langkah 4.4: Tambahkan kode berikut pada **User*Adapter*.kt:**

UserAdapter.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik5

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.kurniawan.belajarandroidkotlin.R

class FruitAdapter(private val fruitList: List<User>) :
    RecyclerView.Adapter<FruitAdapter.FruitViewHolder>() {

    class FruitViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        val userImage: ImageView = itemView.findViewById(R.id.ivUser)
        val userName: TextView = itemView.findViewById(R.id.tvUserName)
    }

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): FruitViewHolder {
        val view = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_user, parent, attachToRoot: false)
        return FruitViewHolder(view)
    }

    override fun onBindViewHolder(holder: FruitViewHolder, position: Int) {
        val user = fruitList[position]
        holder.userImage.setImageResource(user.image)
        holder.userName.text = user.name
    }

    override fun getItemCount(): Int = fruitList.size
}
```

5. Menambahkan Layout Item

Langkah 5.1*:* Buat layout **item_user.xml** di dalam folder **res/layout/** dengan cara klik kanan pada package ***layout***. Yang berfungsi untuk menampilkan item daftar pengguna.

Langkah 5.2: Tambahkan kode berikut pada **item_user.xml**:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
```

```
"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:padding="8dp"
 android:gravity="center_vertical">

 <ImageView
 android:id="@+id/ivUser"
 android:layout_width="40dp"
 android:layout_height="40dp"
 android:layout_marginEnd="8dp"
 android:src="@drawable/ic_launcher_foreground" />

 <TextView
 android:id="@+id/tvUserName"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Nama"
 android:textSize="18sp" />
</LinearLayout>
```

6. **Mengkonfigurasi AndroidManifest.xml**
   Langkah 6.1: Tambahkan deklarasi setaip *Activity* di **AndroidManifest.xml** untuk mengenali file-file ini dalam aplikasi:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BelajarAndroidKotlin"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        tools:targetApi="31">
        <activity
            android:name=".topik5.RecycleActivity"
            android:exported="false" />
        <activity
```

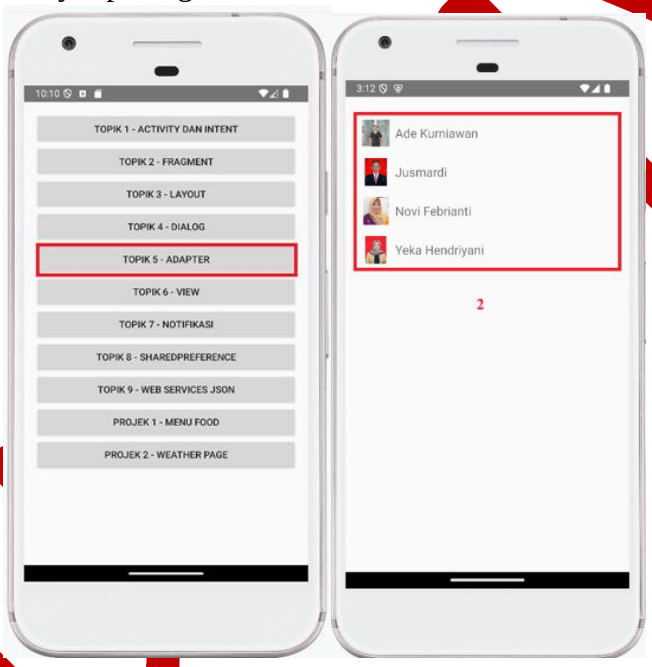
7. **Menjalankan Dan Menguji Aplikasi**
   Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.
   Langkah 7.1: Klik **Run** pada toolbar Android Studio atau tekan **Shift + F10** untuk memulai proses kompilasi dan menjalankan aplikasi.

Langkah 7.2: Pada jendela **Select Deployment Target**, pilih perangkat yang akan digunakan untuk menguji aplikasi:

a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode **USB Debugging** aktif.

8. Hasil Pengujian

Setelah aplikasi berhasil dijalankan, Anda akan melihat tampilan berikut pada layar perangkat atau emulator:



Gambar 12. Hasil Pengujian Adapter dan Recyle View

D. Tugas

1. **Modifikasi:** Setelah menyelesaikan penggunaan *Adapter* dan *RecyclerView* di atas lanjutkan modifikasi dengan membuat contoh list lain seperti list nama anggota kelas atau list musik.
2. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan

DRAFT

BAB 8
VIEW

Pada bab ini, kita akan membahas berbagai komponen penting yang berperan dalam meningkatkan interaksi pengguna, fungsionalitas, dan performa aplikasi Android. Sebagai platform yang sangat fleksibel dan populer, Android menawarkan berbagai komponen antarmuka pengguna (UI) dan fitur yang dapat membantu pengembang menciptakan aplikasi yang intuitif dan efisien. Beberapa komponen yang akan dibahas di antaranya adalah **SearchView, *WebView*, TabLayout, NestedScrollView, dan CardView,** yang masing-masing memiliki kegunaan khusus dalam menyusun antarmuka aplikasi Android yang lebih interaktif dan mudah digunakan.

SearchView memudahkan pengguna untuk melakukan pencarian dalam aplikasi, sementara ***WebView*** memungkinkan aplikasi untuk menampilkan halaman web langsung di dalam antarmuka tanpa perlu membuka browser eksternal. Untuk aplikasi yang membutuhkan navigasi antar berbagai bagian, **TabLayout** dan **ViewPager** memungkinkan pengguna untuk beralih antar tab dengan gesekan (swipe) yang halus dan responsif. Di sisi lain, **NestedScrollView** dan **CardView** memberikan fleksibilitas dalam pengaturan layout, memungkinkan aplikasi menampilkan konten yang panjang atau terstruktur dalam bentuk yang menarik dan mudah diakses.

Melalui implementasi yang tepat dari komponen-komponen ini, pengembang dapat meningkatkan pengalaman pengguna, mengoptimalkan penggunaan ruang pada layar, dan memperkenalkan fitur-fitur canggih dengan lebih efisien. Oleh karena itu, pemahaman yang mendalam tentang cara kerja dan penerapan setiap komponen ini sangat penting untuk menciptakan aplikasi Android yang efektif dan menarik.

A. SearchView

SearchView adalah salah satu widget dalam Android yang sangat bermanfaat untuk menyediakan fitur pencarian di dalam aplikasi. SearchView memungkinkan pengguna memasukkan kueri pencarian dan menampilkan hasilnya secara dinamis. Widget ini biasanya ditempatkan di toolbar atau pada bagian atas layar aplikasi untuk memberikan akses yang mudah kepada pengguna. Implementasi SearchView sangat penting dalam aplikasi yang memiliki banyak data atau konten, seperti aplikasi e-

DUMMY

commerce, berita, dan media sosial, di mana pengguna sering membutuhkan cara cepat untuk menemukan informasi spesifik.

1. Kegunaan SearchView

   Dalam pengembangan aplikasi modern, kemudahan pengguna dalam menavigasi dan menemukan konten menjadi aspek penting dari pengalaman pengguna (user experience). SearchView memungkinkan pengguna melakukan pencarian dengan mengetikkan kueri yang relevan. Setelah itu, aplikasi dapat memperbarui tampilan untuk menampilkan hasil yang sesuai. Sebagai contoh, dalam aplikasi belanja, pengguna dapat mengetikkan "sepatu" di SearchView untuk melihat berbagai produk terkait tanpa harus menjelajahi seluruh katalog.

2. Implementasi SearchView dalam *Fragment*

   Fragment adalah bagian yang dapat diintegrasikan ke dalam satu atau beberapa *Activity* dalam Android. *Fragment* memiliki siklus hidup (lifecycle) tersendiri, yang memungkinkan pengembang untuk mengelola tampilan dan interaksi pengguna pada berbagai bagian aplikasi. Dalam konteks *Fragment*, SearchView dapat diintegrasikan dengan lifecycle dari *Fragment*, sehingga kueri pengguna dapat dikelola di setiap instance *Fragment* yang sedang aktif.

   a. Inisialisasi SearchView di *Fragment*

      Untuk menginisialisasi SearchView di dalam *Fragment*, pengembang dapat menambahkan komponen SearchView pada layout XML *Fragment* atau langsung melalui kode Java/Kotlin dalam class *Fragment*.

      Contohnya, di file XML *Fragment*, SearchView dapat didefinisikan sebagai berikut:Ketik konten Sub Bab disini

```
<androidx.appcompat.widget.SearchView
android:id="@+id/search_view"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:queryHint="Cari di sini..." />
```

   b. Mengatur Listener pada SearchView

      Dalam *Fragment*, kita menggunakan setOnQueryTextListener untuk memantau perubahan teks yang dimasukkan oleh pengguna. Listener ini terdiri dari dua metode utama:

DUMMY

1) **onQueryTextSubmit:** Memproses kueri pengguna ketika tombol pencarian ditekan. Biasanya, metode ini digunakan ketika pengguna selesai memasukkan kueri dan siap melihat hasil akhir.
2) **onQueryTextChange:** Mengawasi setiap perubahan teks saat pengguna mengetik. Metode ini berguna untuk **memberikan** saran pencarian atau pembaruan hasil secara langsung (real-time) berdasarkan kueri pengguna.

Implementasi **setOnQueryTextListener** dalam *Fragment* dapat dilakukan sebagai berikut:

```
val searchView = view.findViewById<SearchView>(R.id.search_view)
searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {
 override fun onQueryTextSubmit(query: String?): Boolean {
 // Proses kueri akhir yang dimasukkan pengguna
 query?.let {
  // Misalnya, panggil fungsi untuk menampilkan hasil pencarian
  tampilkanHasilPencarian(it)
 }
 return true
 }

 override fun onQueryTextChange(newText: String?): Boolean {
 // Perbarui saran atau hasil pencarian secara langsung
 newText?.let {
  // Misalnya, panggil fungsi untuk memperbarui daftar pencarian
  perbaruiHasilPencarian(it)
 }
 return true
 }
})
```

c. Pengelolaan Kueri Pengguna di *Fragment*

Salah satu keunggulan mengimplementasikan SearchView di *Fragment* adalah kemampuan untuk mengelola hasil pencarian pada setiap instance *Fragment* yang aktif. Ketika pengguna berpindah antar *Fragment* atau membuka *Fragment* lain, *Fragment* dapat mempertahankan hasil pencarian atau saran yang relevan sesuai konteks pencarian saat itu.

Ini juga memungkinkan *Fragment* untuk menyimpan dan mengelola status pencarian pengguna selama siklus hidup *Fragment*. Sebagai contoh, ketika *Fragment* di restart atau dirotasi, status pencarian dapat dikembalikan dengan menggunakan **savedInstanceState.**

d. Menampilkan Hasil Pencarian secara Dinamis
   Setelah kueri ditangkap oleh SearchView, langkah selanjutnya adalah memperbarui tampilan hasil sesuai dengan kueri. Pada aplikasi e-commerce, misalnya, kueri dari SearchView dapat dikirim ke API atau database internal untuk membuat produk-produk yang cocok dengan kueri tersebut. Hasil ini kemudian ditampilkan menggunakan *RecyclerView* atau ListView yang terintegrasi dengan *Fragment*.
e. Manfaat Penggunaan SearchView dalam *Fragment*
   1) Integrasi SearchView dengan *Fragment* memberikan fleksibilitas yang tinggi dalam hal pemeliharaan dan pengelolaan hasil pencarian.
   2) *Fragment* dapat mengelola kueri dan hasil pencarian secara mandiri, yang sangat bermanfaat pada aplikasi yang membutuhkan navigasi antar bagian yang kompleks.
   3) Penggunaan listener seperti **onQueryTextChange** juga memungkinkan pencarian prediktif, di mana hasil diperbarui saat pengguna mengetik. Fitur ini meningkatkan kenyamanan pengguna dalam menavigasi aplikasi dan membantu mereka menemukan konten dengan cepat.

B. *WebView*

WebView adalah komponen tampilan yang memungkinkan aplikasi Android untuk membuat dan menampilkan halaman web atau konten berbasis HTML secara langsung di dalam antarmuka aplikasi. Komponen ini sangat bermanfaat bagi aplikasi yang perlu menampilkan konten web tanpa harus membuka browser eksternal, yang dapat mengganggu pengalaman pengguna. *WebView* sering digunakan dalam aplikasi berita, e-commerce, dan sosial media, di mana terdapat kebutuhan untuk menampilkan informasi atau layanan eksternal dari web, namun tetap ingin menjaga pengguna berada di dalam aplikasi.

1. Cara Kerja *WebView*
   WebView didukung oleh WebKit engine, yang merupakan mesin rendering yang digunakan untuk menampilkan dan mengelola konten

DUMMY

web. Dengan menggunakan WebKit engine, *WebView* dapat merender berbagai jenis konten web, termasuk HTML, CSS, dan JavaScript.

Saat *WebView* diberi URL untuk dimuat, komponen ini secara otomatis mengirim permintaan ke server dan menampilkan respons yang diterima dalam format halaman web. *WebView* dapat menangani beberapa fungsi dasar peramban, termasuk:

2. Back dan Forward

   WebView memiliki kemampuan untuk navigasi mundur (back) atau maju (forward) di antara halaman-halaman web yang sudah dibuka sebelumnya, mirip dengan fungsi di peramban.
3. Refresh

   WebView juga mendukung fungsi refresh untuk memuat ulang halaman web yang ditampilkan.
4. JavaScript

   Secara default, JavaScript tidak diaktifkan pada *WebView*, namun pengembang dapat mengaktifkannya menggunakan getSettings().setJavaScriptEnabled(true) untuk mendukung halaman-halaman yang memerlukan JavaScript.
5. Manfaat Penggunaan *WebView*

   a. Integrasi Konten Web dalam Aplikasi

      WebView memungkinkan aplikasi untuk memuat konten eksternal seperti artikel berita, halaman produk, atau bahkan aplikasi web. Ini memberikan fleksibilitas bagi pengembang untuk menambahkan fitur atau konten dari situs web pihak ketiga tanpa memerlukan pengguna keluar dari aplikasi.

   b. Penghematan Waktu dan Biaya Pengembangan

      Dalam beberapa kasus, menggunakan *WebView* untuk menampilkan konten web lebih efisien daripada mengembangkan konten atau antarmuka khusus di dalam aplikasi. Misalnya, jika aplikasi e-commerce memiliki halaman FAQ di situs web, halaman tersebut dapat langsung dimuat melalui *WebView*, menghemat waktu pengembangan yang dibutuhkan untuk membuat ulang halaman dalam aplikasi.

   c. Pengalaman Pengguna yang Konsisten

      Dengan memuat konten web di dalam aplikasi, *WebView* membantu menjaga pengalaman pengguna yang konsisten. Pengguna tidak perlu dialihkan ke browser eksternal, yang dapat mengganggu alur aplikasi dan menurunkan tingkat retensi pengguna.

DUMMY

6. Implementasi *WebView* di Aplikasi Android
   Berikut ini adalah langkah-langkah untuk mengimplementasikan *WebView* di dalam aplikasi Android. Dalam contoh ini, *WebView* digunakan untuk menampilkan halaman web dari URL yang diberikan.
   a. Penambahan *WebView* di Layout XML
      Pertama, tambahkan elemen *WebView* ke dalam layout XML. Elemen ini akan menjadi tempat tampilan dari konten web yang dimuat.

```
<WebView
android:id="@+id/web_view"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

   b. Inisialisasi dan Konfigurasi *WebView* di Kode Program
      Inisialisasi *WebView* dalam kode Java atau Kotlin, dan atur berbagai pengaturan yang dibutuhkan, seperti mengaktifkan JavaScript jika halaman memerlukannya

```
val WebView = findViewById<WebView>(R.id.web_view)

// Mengaktifkan JavaScript jika halaman memerlukan
WebView.settings.javaScriptEnabled = true

// Menentukan URL yang akan dimuat
WebView.loadUrl("https://www.example.com")
```

   c. Mengaktifkan Fungsi Navigasi dalam *WebView*
      Untuk menyediakan pengalaman navigasi yang lebih kaya, *WebView* dapat disertai dengan fitur navigasi seperti tombol back atau forward. Berikut ini contoh cara menangani tombol back dalam *WebView*:

```
override fun onBackPressed() {
 if (WebView.canGoBack()) {
 // Jika bisa kembali ke halaman sebelumnya
 WebView.goBack()
 } else {
 // Jika tidak, keluar dari WebView
 super.onBackPressed()
 }
}
```

   d. Penanganan Pengalihan (Override URL Loading)

Secara default, ketika pengguna menekan tautan di dalam *WebView*, tautan tersebut dapat terbuka di peramban eksternal. Agar tetap berada di dalam *WebView*, kita perlu mengatur metode *WebView*Client dan shouldOverrideUrlLoading:

```
WebView.WebViewClient = object : WebViewClient() {
 override fun shouldOverrideUrlLoading(view: WebView?,
url: String?): Boolean {
 url?.let { view?.loadUrl(it) }
 return true
 }
}
```

C. TabHost dengan TabLayout

TabHost adalah salah satu komponen penting dalam pengembangan aplikasi Android yang berfungsi sebagai wadah atau penampung bagi kumpulan tab yang memungkinkan pengguna mengakses berbagai fitur atau informasi terkait dalam satu tampilan. TabHost sering digunakan ketika aplikasi memerlukan pengelompokan informasi atau fitur-fitur yang terstruktur, sehingga pengguna dapat menavigasi antara bagian-bagian yang berbeda tanpa harus berpindah dari satu layar ke layar lain. Dalam implementasi modern, TabLayout biasanya digunakan bersama ViewPager untuk menciptakan pengalaman pengguna yang lebih interaktif dan responsif dengan menggunakan navigasi berbasis gesekan (swipe).

TabHost dan TabLayout sangat berguna dalam membuat aplikasi yang kaya fitur namun tetap terstruktur dan mudah dinavigasi. Penggunaan tab memungkinkan pengguna untuk mengakses fitur yang berbeda atau data yang dikelompokkan secara logis tanpa harus bolak-balik ke layar utama. TabLayout bersama ViewPager juga memungkinkan navigasi yang lancar dan mulus, dengan tampilan yang konsisten dan responsif. Implementasi ini sangat sesuai untuk aplikasi berita, media sosial, keuangan, dan jenis aplikasi lainnya yang membutuhkan pengelompokan data atau fitur.

1. Tahapan Implementasi TabHost - TabLayout dengan ViewPager

   Berikut adalah langkah-langkah terperinci untuk mengimplementasikan TabHost bersama TabLayout menggunakan ViewPager di dalam aplikasi Android:

   a. Menambahkan Dependensi pada Project

      Sebelum memulai implementasi, pastikan Anda telah menambahkan dependensi yang diperlukan untuk mendukung TabLayout dan

DUMMY

ViewPager. Biasanya, dependensi ini adalah bagian dari library AndroidX atau Material Components.

Berikut adalah contoh penambahan dependensi di file *Build*. *Gradle* :

```
dependencies {
 implementation
'com.google.android.material:material:1.4.0'
 implementation 'androidx.viewpager2:viewpager2:1.0.0'
}
```

b. Pembuatan XML Layout untuk TabLayout dan ViewPager

Langkah berikutnya adalah mendefinisikan TabLayout dan ViewPager di file layout XML, yang akan menjadi tempat tampilan dari setiap tab dan halaman yang akan diakses melalui swipe.

Contoh layout XML untuk TabLayout dan ViewPager adalah sebagai berikut:

```
<com.google.android.material.tabs.TabLayout
android:id="@+id/tab_layout"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>

<androidx.viewpager2.widget.ViewPager2
android:id="@+id/view_pager"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

TabLayout biasanya ditempatkan di bagian atas layar, sedangkan ViewPager berada di bawahnya untuk menampilkan isi dari setiap tab.

c. Membuat ViewPager dan Mengkonfigurasinya dengan *Adapter*

ViewPager adalah elemen utama yang memungkinkan pengguna menggulir antar halaman atau tab. ViewPager bekerja dengan bantuan ***Adapter*** yang menyediakan *Fragment* untuk setiap halaman/tab.

Pertama, buat kelas *Adapter* yang mengelola *Fragment* untuk setiap tab. Dalam contoh ini, kita akan membuat *Adapter* bernama **ViewPager*Adapter*** yang memperluas ***Fragment*StateAdapter.**

```
class ViewPagerAdapter(Fragment Activity: Fragment
Activity) : FragmentStateAdapter(Fragment Activity) {
 private val FragmentList = ArrayList<Fragment>()
 private val FragmentTitleList = ArrayList<String>()

 fun addFragment(Fragment: Fragment, title: String) {
```

DUMMY

```
 FragmentList.add(Fragment)
 FragmentTitleList.add(title)
 }

 override fun getItemCount(): Int {
 return FragmentList.size
 }

 override fun createFragment(position: Int): Fragment {
 return FragmentList[position]
 }
}
```

Setelah *Adapter* dibuat, kita dapat menghubungkan *Adapter* ini dengan ViewPager di *Activity* atau *Fragment* utama.

```
val viewPagerAdapter = ViewPagerAdapter(this)
viewPager.Adapter = viewPagerAdapter
```

d. Instansi *Fragment* untuk Setiap Tab

Untuk setiap tab, kita perlu membuat *Fragment* terpisah yang akan ditampilkan sesuai dengan tab yang dipilih. Setiap *Fragment* dapat memiliki konten atau fitur yang berbeda, sesuai kebutuhan aplikasi. Misalnya, dalam aplikasi manajemen keuangan, *Fragment* pertama mungkin menampilkan "Pengeluaran," *Fragment* kedua menampilkan "Pemasukan," dan *Fragment* ketiga menampilkan "Laporan Keuangan."

Setelah *Fragment* untuk setiap tab dibuat, kita tambahkan *Fragment* ini ke *Adapter* menggunakan metode add*Fragment* yang telah dibuat sebelumnya.

```
viewPagerAdapter.addFragment(ExpenseFragment(),
"Pengeluaran")
viewPagerAdapter.addFragment(IncomeFragment(),
"Pemasukan")
viewPagerAdapter.addFragment(ReportFragment(),
"Laporan")
```

e. Menghubungkan TabLayout dengan ViewPager

Langkah terakhir adalah menghubungkan TabLayout dengan ViewPager, sehingga setiap tab dapat digulir atau dipilih secara manual oleh pengguna. Untuk melakukannya, kita dapat menggunakan TabLayoutMediator, yang akan secara otomatis mengikat ViewPager ke TabLayout.

```
TabLayoutMediator(tabLayout, viewPager) { tab,
```

DRAFT

```
position ->
 tab.text                                                    =
viewPager.Fragment.TitleList[position]
}.attach()
```

Dengan ini, setiap tab di TabLayout akan disinkronkan dengan halaman yang ditampilkan di ViewPager. Pengguna dapat menggulir antara tab dengan gesekan (swipe) atau dengan mengetuk tab yang diinginkan.

2. Keuntungan Penggunaan TabHost dengan TabLayout
   a. Pengalaman Pengguna yang Mulus dan Interaktif
      TabHost dengan TabLayout memberikan pengalaman pengguna yang lebih baik dengan memungkinkan navigasi yang mulus antara berbagai bagian aplikasi. Pengguna dapat menggulir antara tab dengan cepat tanpa harus memuat ulang data atau pindah ke layar lain.
   b. Penyusunan Informasi yang Lebih Baik
      Tab memungkinkan aplikasi untuk mengelompokkan informasi atau fitur berdasarkan kategori tertentu, yang membuat aplikasi lebih terstruktur dan lebih mudah dipahami oleh pengguna. Misalnya, dalam aplikasi media sosial, tab dapat digunakan untuk membagi konten menjadi "Beranda," "Notifikasi," dan "Profil."
   c. Navigasi Berbasis Gesekan (Swipe)
      Integrasi TabLayout dengan ViewPager menyediakan fitur navigasi berbasis gesekan, yang sangat intuitif bagi pengguna. Ini membuat aplikasi terasa lebih modern dan sesuai dengan ekspektasi pengguna saat ini.
   d. Pengelolaan *Fragment* yang Mudah
      Dengan menggunakan *Fragment* untuk setiap tab, pengembang dapat mengelola dan mengatur ulang fitur atau tampilan yang diinginkan dengan mudah. *Fragment* terpisah untuk setiap tab juga memungkinkan pembaruan atau penyesuaian fitur tanpa mempengaruhi tab lainnya.
   e. Dukungan untuk Kustomisasi dan Gaya yang Konsisten
      TabLayout menyediakan opsi untuk kustomisasi tampilan tab, seperti mengganti warna teks, ikon, atau indikator tab. Ini memungkinkan pengembang menciptakan tampilan yang konsisten sesuai dengan gaya visual aplikasi.

D. NestedScrollView dan Cardview

DUMMY

NestedScrollView dan CardView adalah dua komponen antarmuka yang umum digunakan dalam pengembangan aplikasi Android untuk memberikan tata letak yang lebih terstruktur, menarik, dan nyaman diakses oleh pengguna. NestedScrollView merupakan komponen scroll yang memungkinkan pengguna menggulir konten yang panjang, bahkan di dalamnya bisa berisi komponen-komponen scroll lainnya. Di sisi lain, CardView adalah komponen tampilan yang digunakan sebagai wadah (container) untuk elemen-elemen visual dalam bentuk kartu, membantu menyusun informasi dalam tampilan yang lebih terfokus dan menarik.

Penggunaan NestedScrollView dan CardView secara bersamaan dalam aplikasi memungkinkan pengembang untuk menyajikan konten dalam bentuk yang elegan dan mudah dibaca. Contoh penggunaan umum NestedScrollView dan CardView meliputi aplikasi berita, media sosial, dan e-commerce, di mana banyak informasi atau produk perlu ditampilkan dalam tata letak yang rapi dan mudah digulir.

1. Fungsi dan Manfaat NestedScrollView
   a. Kemampuan Menggulung Konten Panjang
      NestedScrollView memungkinkan pengguna untuk menggulir area konten yang panjang tanpa batasan pada jumlah atau ukuran elemen yang ditampilkan. Ini sangat berguna untuk aplikasi yang memuat banyak konten atau informasi dalam satu layar, seperti artikel berita atau feed media sosial.
   b. Dukungan untuk Scroll di dalam Scroll
      Salah satu kelebihan utama NestedScrollView adalah kemampuannya untuk berisi komponen scroll lain, misalnya *RecyclerView* atau HorizontalScrollView. Fitur ini memungkinkan pengguna untuk menggulir secara vertikal di dalam NestedScrollView, sekaligus menggulir secara horizontal pada komponen di dalamnya. Ini memberikan pengalaman pengguna yang lebih fleksibel.
   c. Kompatibilitas dengan Elemen UI Lainnya
      NestedScrollView mendukung berbagai elemen UI dan memungkinkan pengguna untuk memasukkan komponen-komponen seperti Button, ImageView, TextView, dan CardView di dalamnya. Ini memberikan fleksibilitas dalam membuat tampilan yang beragam dan menarik.
2. Fungsi dan Manfaat NestedScrollView
   a. Menampilkan Konten dalam Bentuk Kartu

DUMMY

CardView digunakan untuk menampilkan elemen-elemen konten dalam bentuk kartu dengan gaya yang lebih fokus dan terorganisir. Kartu ini bisa berupa gambar, teks, atau komponen lain yang ingin ditonjolkan. Tampilan ini memudahkan pengguna untuk fokus pada setiap bagian konten yang disajikan.

b. Efek Bayangan dan Sudut Membulat
CardView memiliki fitur bawaan untuk menambahkan bayangan dan sudut yang membulat pada tepinya. Efek visual ini memberikan kesan bahwa elemen tersebut "mengambang" di atas latar belakang, membuat tampilan lebih menarik secara visual. Dengan CardView, desain aplikasi terasa lebih modern dan profesional.

c. Pemfokusan Konten
CardView sering digunakan sebagai kontainer untuk elemen-elemen konten yang perlu difokuskan, seperti profil pengguna, deskripsi produk, atau kartu artikel. Dengan menggunakan CardView, informasi di dalam aplikasi dapat ditampilkan secara lebih tertata dan terfokus, sehingga memudahkan pengguna untuk memahami konten yang disampaikan.

3. Implementasi NestedScrollView dengan CardView
Berikut adalah langkah-langkah implementasi NestedScrollView dan CardView dalam kode XML untuk memberikan pengalaman pengguna yang lebih baik.

a. Membuat NestedScrollView di Layout XML:
NestedScrollView ditempatkan di dalam layout XML untuk mencakup area scroll utama. NestedScrollView memungkinkan konten yang panjang digulirkan di dalam aplikasi, dan di dalamnya dapat terdapat komponen-komponen scroll lainnya jika dibutuhkan.

```
<androidx.core.widget.NestedScrollView
android:id="@+id/nested_scroll_view"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fillViewport="true">

<!-- Konten NestedScrollView ditempatkan di sini -->
</androidx.core.widget.NestedScrollView>
```

Parameter fillViewport="true" digunakan untuk memastikan bahwa NestedScrollView memenuhi seluruh layar jika konten di dalamnya lebih sedikit dari layar, sehingga memberikan tampilan yang konsisten.

DRAFT

b. Menambahkan CardView ke dalam NestedScrollView:
Setelah NestedScrollView didefinisikan, kita dapat menambahkan CardView di dalamnya sebagai wadah untuk konten yang ingin ditonjolkan. CardView memungkinkan elemen-elemen konten disajikan dengan tampilan yang lebih menarik, seperti informasi produk atau berita dalam bentuk kartu.

```
<androidx.core.widget.NestedScrollView
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fillViewport="true">

<androidx.cardview.widget.CardView
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:layout_margin="16dp"
 app:cardCornerRadius="8dp"
 app:cardElevation="4dp">

 <!-- Isi dari CardView, misalnya TextView atau
ImageView -->
 <LinearLayout
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="vertical"
 android:padding="16dp">

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Judul Konten"
  android:textSize="18sp"
  android:textStyle="bold"/>

 <TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="Deskripsi singkat atau konten tambahan
yang ingin disajikan."
  android:textSize="14sp"/>

 </LinearLayout>

</androidx.cardview.widget.CardView>

</androidx.core.widget.NestedScrollView>
```

Pada contoh di atas, CardView diletakkan di dalam NestedScrollView. CardView ini memiliki margin di sekelilingnya (`android:layout_margin="16dp"`), radius sudut (`app:cardCornerRadius="8dp"`), dan elevasi (`app:cardElevation="4dp"`) untuk memberikan efek bayangan. LinearLayout di dalam CardView berfungsi sebagai kontainer untuk elemen-elemen konten, seperti TextView yang menampilkan judul dan deskripsi.

c. Menambahkan CardView ke dalam NestedScrollView:
NestedScrollView mendukung berbagai pengaturan tambahan untuk memastikan konten dapat digulir secara halus. Pengaturan fillViewport="true" pada NestedScrollView memastikan bahwa konten akan memenuhi layar sepenuhnya, bahkan jika konten tidak terlalu panjang.
CardView juga bisa diatur lebih lanjut, seperti menambahkan `app:cardUseCompatPadding="true"` untuk kompatibilitas padding yang lebih baik di berbagai versi Android.

E. Implementasi View

Langkah - langkah implementasi pada aplikasi

1. Membuat Class Kotlin
Langkah 1.1: Buka kembali project Belajar Kotlin yang telah di buat pada Job Sheet sebelumnya:
Langkah 1.2: Perbarui kode pada Main *Activity*.kt

```
val button5 = findViewById<Button>(R.id.button5)
button5.setOnClickListener {
    val intent = Intent( packageContext: this, MenuView::class.java)
    startActivity(intent)
}
```

Langkah 1.3: Buat *Activity* baru dengan nama MenuView.kt dan buatlah kode seperti di bawah berikut :

DRAFT

MenuView.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.jobsheet8View

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R

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

        val buttonSearchView = findViewById<Button>(R.id.button_to_searchview)
        buttonSearchView.setOnClickListener{
            // Pastikan bahwa Anda memiliki Activity bernama SearchViewActivity
            val intent = Intent( packageContext: this, SearchView::class.java)
            startActivity(intent)
        }

        val buttonTabHost = findViewById<Button>(R.id.button_to_tabhost)
        buttonTabHost.setOnClickListener{
            // Pastikan bahwa Anda memiliki Activity bernama TabHostActivity
            val intent = Intent( packageContext: this, TabHost::class.java)
            startActivity(intent)
        }
    }
}
```

Langkah 1.4: Buat *Activity* baru dengan nama SearchView.kt dan buatlah kode seperti di bawah berikut :

DUMMY

SearchView.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.jobsheet8View

import android.os.Bundle
import android.widget.SearchView as AndroidSearchView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R

class SearchView : AppCompatActivity() { // Mengganti nama kelas untuk menghindari konflik

    private val userList = listOf(
        User( name: "Galant Anefsyah Pratama",  email: "galantpratama@gmail.com"),
        User( name: "Muhammad Irzan Ali",  email: "adan@gmail.com"),
        User( name: "Salman Rizky",  email: "salman@gmail.com")
    )

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_search_view) // Pastikan layout ini sesuai

        val searchView = findViewById<AndroidSearchView>(R.id.searchView) // Menggunakan alias untuk menghindari bentrok nama
        val resultTextView = findViewById<TextView>(R.id.resultTextView) // TextView untuk menampilkan hasil pencarian

        // Logika pencarian di SearchView
        searchView.setOnQueryTextListener(object : AndroidSearchView.OnQueryTextListener {
            override fun onQueryTextSubmit(query: String?): Boolean {
                return false
            }

            override fun onQueryTextChange(newText: String?): Boolean {
                filterUsers(newText.orEmpty(), resultTextView) // Mengupdate hasil pencarian
                return true
            }
        })
    }

    private fun filterUsers(query: String, resultTextView: TextView) {
        // Filter pengguna berdasarkan nama
        val filteredUsers = userList.filter { it.name.contains(query, ignoreCase = true) }
```

Langkah 1.5: Buat *Activity* baru dengan nama TabHost.kt dan buatlah kode seperti di bawah berikut :

DUMMY

TabHost.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.jobsheet8View

import android.os.Bundle
import android.widget.ArrayAdapter
import android.widget.ListView
import android.widget.TabHost
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R

class TabHost : AppCompatActivity() {

    private lateinit var listView: ListView
    private lateinit var arrayAdapter: ArrayAdapter<String>
    private val userList = listOf("John", "Jane", "Jack", "Jessica", "James", "Julia")

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_tab_host)

        val tabHost = findViewById<TabHost>(R.id.tabHost)
        tabHost.setup()

        // Menambahkan Tab 1
        val tabSpec1 = tabHost.newTabSpec( tag: "Tab Satu")
        tabSpec1.setContent(R.id.tab1)  // Pastikan R.id.tab1 benar ada di layout XML
        tabSpec1.setIndicator("Daftar Pengguna")
        tabHost.addTab(tabSpec1)

        // Menambahkan Tab 2
        val tabSpec2 = tabHost.newTabSpec( tag: "Tab Dua")
        tabSpec2.setContent(R.id.tab2)  // Pastikan R.id.tab2 benar ada di layout XML
        tabSpec2.setIndicator("Tab 2")
        tabHost.addTab(tabSpec2)

        // Inisialisasi ListView
        listView = findViewById(R.id.listView)
```

Langkah 1.6: Buat Modul User dan Isikan script berikut :

TabHost.kt | MainActivity.kt | activity_main.xml | User.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.jobsheet8View

data class User(val name: String, val email: String)
```

2. Menambahkan Komponen UI

Langkah 2.1: Isikan kode xml ke *Activity*_menu_view seperti berikut:

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

```
android:orientation="vertical"
android:layout_marginTop="24dp"
>
<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="Belajar View"
android:layout_marginTop="10dp"
android:layout_marginLeft="10dp"
android:textSize="15dp"
android:textFontWeight="800"
android:gravity="center"
>
</TextView>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
>
<Button
 android:id="@+id/button_to_searchview"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="Search View" />
</LinearLayout>

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="10dp"
>
<Button
 android:id="@+id/button_to_tabhost"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_centerInParent="true"
 android:text="Tab Host" />
</LinearLayout>
</LinearLayout>
```

Langkah 2.2: Buatlah xml baru dengan nama actvitity_tab_host dan isikan kode berikut:

```
<TabHost
xmlns:android="http://schemas.android.com/apk/res/andro
id"
```

```
    android:id="@+id/tabHost"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TabWidget
     android:id="@android:id/tabs"
     android:layout_width="match_parent"
     android:layout_height="wrap_content" />

    <FrameLayout
     android:id="@android:id/tabcontent"
     android:layout_width="match_parent"
     android:layout_height="match_parent">

     <!-- Konten untuk Tab 1 -->
     <LinearLayout
     android:id="@+id/tab1"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="vertical">

     <ListView
      android:id="@+id/listView"
      android:layout_width="match_parent"
      android:layout_height="match_parent" />
     </LinearLayout>

     <!-- Konten untuk Tab 2 -->
     <LinearLayout
     android:id="@+id/tab2"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:orientation="vertical">
     <!-- Tambahkan konten Tab 2 di sini -->
     </LinearLayout>

    </FrameLayout>
    </LinearLayout>
</TabHost>
```

3. **Pengujian Aplikasi**

   Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.

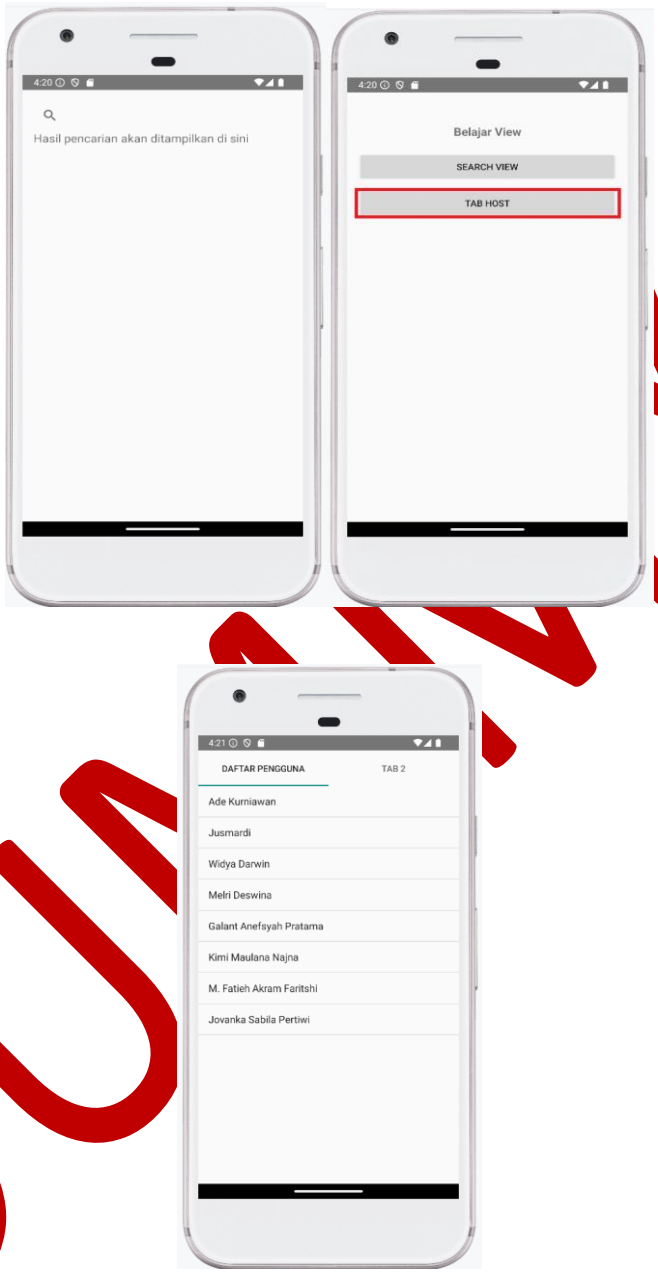
   Langkah 3.1: Klik **Run** pada toolbar Android Studio atau tekan **Shift + F10** untuk memulai proses kompilasi dan menjalankan aplikasi.

   Langkah 3.2: Pada jendela **Select Deployment Target**, pilih perangkat yang akan digunakan untuk menguji aplikasi:

   a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
   b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode USB Debugging aktif.

4. **Hasil Pengujian**

DRAFT



Gambar 13. Hasil Pengujian View

F. Tugas

1. **Modifikasi**: Setelah menyelesaikan langkah-langkah praktikum di atas lanjutkan modifikasi membuat search yang dapat menampilkan data yang ditampilkan pada sebuah *Adapter*. Pedomani kembali pembuatan *Adapter* dan gabungkan dengan search.

DUMMY

2. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.Ketik konten Sub Bab disini

DUMMY

BAB 9
NOTIFIKASI ANDROID

Dalam pengembangan aplikasi Android, salah satu fitur penting yang dapat meningkatkan interaksi dan keterlibatan pengguna adalah notifikasi. Notifikasi memungkinkan aplikasi untuk memberikan informasi atau pengingat kepada pengguna, bahkan ketika aplikasi tidak aktif atau berjalan di latar belakang. Fitur ini berfungsi untuk menjaga aplikasi tetap relevan di mata pengguna, memberikan pembaruan penting, atau sekadar menyampaikan informasi yang perlu diperhatikan. Oleh karena itu, notifikasi sangat bermanfaat dalam berbagai jenis aplikasi, mulai dari aplikasi pesan, media sosial, e-commerce, hingga aplikasi pengingat. Memahami cara kerja dan implementasi notifikasi dalam aplikasi Android sangat penting bagi pengembang untuk menciptakan pengalaman pengguna yang lebih baik.

A. Komponen Dasar Notifikasi

Notifikasi merupakan salah satu fitur utama dalam sistem operasi Android yang memungkinkan aplikasi berkomunikasi dengan pengguna, meskipun aplikasi tersebut tidak sedang dibuka. Notifikasi dapat berupa pesan pengingat, informasi penting, atau komunikasi dari pengguna lain. Fungsi utama notifikasi adalah untuk menarik perhatian pengguna tanpa mengganggu interaksi mereka dengan aplikasi lain atau perangkat secara keseluruhan. Dalam konteks Android, notifikasi ditampilkan di luar antarmuka pengguna aplikasi (UI), sehingga pengguna tetap bisa menerima informasi penting meskipun aplikasi tidak aktif. Pengguna juga memiliki kontrol penuh atas notifikasi, termasuk memilih untuk menonaktifkan atau mengizinkan aplikasi mengirimkan notifikasi.

Agar notifikasi dapat efektif dan menyampaikan informasi dengan jelas, terdapat beberapa komponen yang membentuk notifikasi di Android. Komponen-komponen ini bekerja bersama untuk menciptakan pengalaman yang lebih interaktif dan menarik bagi pengguna. Berikut adalah penjelasan lengkap mengenai komponen-komponen utama yang ada dalam notifikasi Android:

1. Icon

   Icon adalah elemen visual pertama yang dilihat oleh pengguna ketika notifikasi muncul. Icon ini mewakili aplikasi atau jenis notifikasi tertentu. Biasanya, icon ini berupa gambar kecil atau simbol yang mencerminkan identitas aplikasi atau memberi petunjuk tentang isi notifikasi. Misalnya, icon pesan teks dapat berupa gambar amplop, sedangkan icon aplikasi e-commerce dapat berupa gambar keranjang

DUMMY

belanja. Icon bertujuan untuk memberikan petunjuk visual yang jelas sehingga pengguna dapat dengan mudah mengenali sumber atau jenis notifikasi yang mereka terima tanpa harus membuka aplikasi tersebut. Fungsi dari icon untuk memberikan identitas visual pada notifikasi dan membantu pengguna mengenali aplikasi atau jenis informasi yang diberikan. Icoon yang menarik dan relevan dapat meningkatkan peluang notifikasi dibaca dan ditanggapi oleh pengguna.

2. Judul dan Pesan

   Judul notifikasi adalah elemen teks yang paling menonjol dan langsung memberi tahu pengguna tentang inti dari notifikasi tersebut. Misalnya, "Pesan Baru" atau "Pembayaran Berhasil". Judul memberikan gambaran umum tentang apa yang akan dibaca pengguna, dan biasanya berisi informasi yang paling penting. Judul berfungsi memberikan konteks utama. Pesan adalah teks yang lebih rinci dan menjelaskan secara lengkap atau lebih mendalam tentang informasi yang ingin disampaikan. Misalnya, "Anda memiliki pesan baru dari John" atau "Pembayaran Anda telah diterima. Terima kasih telah berbelanja!". Pesan ini dapat mencakup detail atau instruksi yang lebih spesifik yang perlu diperhatikan oleh pengguna. Pesan berfungsi menyampaikan informasi lebih rinci.Judul dan pesan sangat penting karena memastikan teks yang digunakan jelas dan langsung ke intinya agar pengguna dapat memahami notifikasi dengan cepat.

3. Tindakan (Action)

   Tindakan adalah tombol atau opsi yang disertakan dalam notifikasi yang memungkinkan pengguna untuk langsung berinteraksi dengan notifikasi tersebut. Misalnya, sebuah notifikasi pesan dapat menyertakan tombol "Balas" atau "Buka Pesan", yang memungkinkan pengguna untuk melakukan aksi tertentu tanpa harus membuka aplikasi sepenuhnya. Tindakan dapat berfungsi untuk membuka aplikasi, memulai aktivitas tertentu, menghapus notifikasi, atau melakukan aksi lainnya sesuai dengan tujuan notifikasi tersebut. Tindakan juga memudahkan pengguna untuk melakukan tindakan langsung tanpa harus menavigasi ke aplikasi dan menambah interaktivitas dengan pengguna dan memungkinkan mereka untuk merespons notifikasi dengan cepat.

4. Saluran Notifikasi (Notification Channel)

   Sejak Android 8.0 (API level 26), saluran notifikasi diperkenalkan untuk memberikan kontrol lebih lanjut kepada pengguna mengenai kategori notifikasi yang mereka terima. Saluran notifikasi

DUMMY

memungkinkan aplikasi untuk mengelompokkan notifikasi berdasarkan jenis atau prioritasnya, seperti notifikasi pesan, pembaruan aplikasi, atau pemberitahuan penting lainnya. Setiap saluran memiliki pengaturan yang berbeda, termasuk suara, getaran, dan prioritas tampilan. Pengguna dapat mengelola preferensi saluran notifikasi mereka, sehingga mereka dapat memilih untuk menerima notifikasi dari saluran tertentu atau mematikannya sesuai kebutuhan. Mengelompokkan notifikasi berdasarkan kategori atau prioritas untuk memudahkan pengelolaan dan pengaturan. Memberikan fleksibilitas kepada pengguna untuk menyesuaikan preferensi mereka terhadap jenis notifikasi yang ingin diterima.Ketik konten Sub Bab disini

B. Izin Notifikasi

Izin untuk mengirimkan notifikasi harus diperoleh dari pengguna. Pada versi Android sebelum Android 13, aplikasi dapat mengirimkan notifikasi tanpa izin eksplisit, tetapi untuk memastikan aplikasi tidak mengganggu pengguna, penting untuk meminta izin atau memberi pengguna pilihan untuk mengatur preferensi notifikasi. Dalam beberapa kasus, aplikasi dapat menampilkan prompt atau dialog untuk memberi tahu pengguna bahwa aplikasi ingin mengirimkan notifikasi.

Penanganan Izin Notifikasi pada Berbagai Versi Android

1. Android 13 dan terbaru: Aplikasi memerlukan izin eksplisit dari pengguna dengan izin POST_NOTIFICATIONS yang harus dideklarasikan pada AndroidManifest.xml. Pengguna akan diminta untuk memberikan izin ketika aplikasi pertama kali mengirimkan notifikasi.
2. Android versi sebelumnya: Aplikasi tidak memerlukan izin eksplisit untuk mengirimkan notifikasi, namun di beberapa versi, pengguna dapat mengatur preferensi notifikasi di pengaturan perangkat.

Panduan yang Disarankan dalam Meminta Izin Notifikasi

1. Jangan langsung meminta izin begitu aplikasi dibuka, beri tahu pengguna dengan jelas kapan dan mengapa izin diperlukan.
2. Pastikan pengguna mengerti manfaat yang mereka dapatkan dari menerima notifikasi, seperti mendapatkan pembaruan penting atau pengingat yang relevan.
3. Sebaiknya beri pengguna kebebasan untuk mengatur notifikasi, seperti mematikan notifikasi untuk kategori tertentu atau menyesuaikan pengaturan lainnya.

DUMMY

C. Pengelolaan Notifikasi

Pengelolaan notifikasi yang efektif memungkinkan aplikasi untuk memberikan informasi yang relevan kepada pengguna tanpa mengganggu pengalaman mereka secara berlebihan. Dengan menggunakan saluran notifikasi, pengguna dapat mengelompokkan jenis-jenis notifikasi, seperti pemberitahuan pengingat, pembaruan berita, atau pesan pribadi, sesuai dengan preferensi mereka. Hal ini memungkinkan pengguna untuk mengelola pengaturan notifikasi dengan lebih fleksibel dan mengurangi gangguan yang tidak diinginkan. Pada bagian ini, akan dibahas tentang pengelolaan notifikasi dalam aplikasi Android, mulai dari pembuatan saluran notifikasi, kustomisasi tampilan notifikasi, hingga manajemen grup dan prioritas notifikasi. Pemahaman tentang pengelolaan notifikasi ini penting untuk menciptakan aplikasi yang tidak hanya informatif tetapi juga menghormati kenyamanan dan preferensi pengguna.

Saluran notifikasi adalah kategori yang digunakan untuk mengelompokkan notifikasi berdasarkan jenis atau kepentingan informasi. Misalnya, aplikasi berita dapat memiliki saluran untuk pembaruan berita utama, sedangkan aplikasi pengingat memiliki saluran untuk notifikasi pengingat jadwal. Saluran notifikasi memungkinkan aplikasi untuk mengelola dan menyampaikan notifikasi secara lebih terstruktur dan sesuai dengan preferensi pengguna.

1. Pembuatan Saluran Notifikasi Baru

   Pada Android 8.0 (Oreo) dan versi yang lebih baru, aplikasi harus menggunakan saluran untuk membuat notifikasi. Saluran ini membantu mengelompokkan notifikasi berdasarkan jenisnya dan memberikan kontrol lebih besar kepada pengguna mengenai pengaturan notifikasi. Pembuatan saluran melibatkan penentuan ID saluran, nama, dan prioritas yang sesuai dengan konteks notifikasi.
2. Konfigurasi Pengaturan Saluran Notifikasi

   Saluran notifikasi memungkinkan pengaturan preferensi tampilan notifikasi, suara, getaran, dan prioritas. Misalnya, saluran dengan prioritas tinggi dapat digunakan untuk notifikasi penting, sementara saluran pengingat dapat menggunakan suara tertentu. Pengaturan ini memberi kontrol lebih besar kepada pengguna, memungkinkan mereka untuk menyesuaikan notifikasi sesuai kenyamanan pribadi.
3. Pembuatan Notifikasi

DUMMY

Notifikasi dibuat menggunakan kelas Notification yang memungkinkan konfigurasi tampilan dan aksi notifikasi. Notifikasi dapat mencakup elemen seperti ikon, teks, suara, atau tombol aksi. Selain itu, notifikasi dapat disesuaikan dengan gambar, suara khusus, atau animasi untuk menarik perhatian pengguna, menyesuaikan dengan informasi yang ingin disampaikan.

4. Kustomisasi Tampilan Notifikasi

   Tampilan notifikasi dapat dikustomisasi dengan elemen visual seperti ikon aplikasi, gambar besar, atau suara khusus. Pengguna juga dapat menambahkan tombol aksi dalam notifikasi yang memungkinkan interaksi langsung, seperti membuka aplikasi atau menghubungi seseorang. Kustomisasi ini bertujuan untuk meningkatkan pengalaman pengguna dan memastikan bahwa notifikasi relevan dan mudah diperhatikan.

5. Penanganan Aksi Notifikasi

   Aksi notifikasi memungkinkan pengguna untuk berinteraksi langsung dengan notifikasi tanpa membuka aplikasi. Misalnya, tombol "Balas" pada notifikasi pesan memungkinkan pengguna membalas pesan langsung dari panel notifikasi. Aksi ini dapat diatur menggunakan PendingIntent, yang memicu aktivitas tertentu saat pengguna memilih aksi tersebut.

6. Pengelompokan Notifikasi

   Grup notifikasi digunakan untuk mengelompokkan beberapa notifikasi terkait dalam satu tampilan yang lebih terorganisir. Ini bermanfaat ketika banyak notifikasi datang dari sumber yang sama, seperti pesan dalam aplikasi perpesanan atau pembaruan dari aplikasi media sosial. Grup ini membantu pengguna untuk melihat semua notifikasi terkait dalam satu tampilan yang lebih mudah dikelola.

7. Manajemen Grup Notifikasi

   Notifikasi dalam grup dapat dikelola dengan mengatur urutannya, memberikan prioritas pada notifikasi tertentu, atau memilih cara menampilkan grup notifikasi tersebut. Misalnya, notifikasi mendesak dapat ditampilkan di bagian atas grup untuk menarik perhatian pengguna. Pengelolaan grup ini memungkinkan aplikasi menampilkan notifikasi yang lebih penting tanpa mengganggu alur penggunaan aplikasi.

8. Pengaturan Prioritas dalam Grup Notifikasi

   Prioritas notifikasi dalam grup mempengaruhi urutan tampilannya. Notifikasi dengan prioritas lebih tinggi akan muncul di

DUMMY

bagian atas grup dan lebih mencolok bagi pengguna. Prioritas ini juga menentukan apakah notifikasi akan muncul sebagai pop-up atau hanya di panel notifikasi. Pengaturan prioritas memastikan notifikasi penting tidak terlewatkan oleh pengguna dan mendapat perhatian yang sesuai.

D. Implementasi Notifikasi Android

Langkah - langkah implementasi pada aplikasi:

1. **Membuat *Activity* File**

   Langkah 1.1: Buka kembali project Belajar Kotlin yang telah di buat pada Job Sheet sebelumnya:

   Langkah 1.2: Perbarui kode pada Main *Activity*.kt

```
val button7 = findViewById<Button>(R.id.button7)
button7.setOnClickListener {
    val intent = Intent( packageContext: this, NotificationActivity::class.java)
    startActivity(intent)
}
```

Langkah 1.3: Membuat *Activity* baru dengan nama Notification *Activity* dan buat kode seperti berikut:

```
package com.kurniawan.belajarandroidkotlin
import android.app.AlarmManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.os. Build
import android.os.Bundle
import android.provider.Settings
import android.util.Log
import android.widget.Button
import android.widget.Toast
import androidx.annotation.RequiresApi
import androidx.appcompat.app.AppCompat Activity
import java.util.*

class Notification Activity : AppCompat Activity() {

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout. Activity_notification)

 if ( Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
  checkAndRequestExactAlarmPermission()
 }
```

```
 val                    buttonSetAlarm                    =
findViewById<Button>(R.id.buttonSetAlarm)
 buttonSetAlarm.setOnClickListener {
  Toast.makeText(this,          "Setting          Alarm...",
Toast.LENGTH_SHORT).show()
  Log.d("Notification   Activity",   "Button   Set   Alarm
clicked")
  setAlarmNotification()
 }

 val                  buttonStartService                  =
findViewById<Button>(R.id.buttonStartService)
 buttonStartService.setOnClickListener {
  val         serviceIntent         =         Intent(this,
NotificationService::class.java)
  startService(serviceIntent)
 }
 }

 @RequiresApi( Build.VERSION_CODES.S)
 private fun checkAndRequestExactAlarmPermission() {
 val                     alarmManager                     =
getSystemService(Context.ALARM_SERVICE) as AlarmManager
 if (!alarmManager.canScheduleExactAlarms()) {
  val                         intent                         =
Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM).apply
{
  data = android.net.Uri.parse("package:$packageName")
  }
  start Activity(intent)
 }
 }

 private fun setAlarmNotification() {
 val                     alarmManager                     =
getSystemService(Context.ALARM_SERVICE) as AlarmManager
 val intent = Intent(this, AlarmReceiver::class.java)
 val pendingIntent = PendingIntent.getBroadcast(
  this,
  0,
  intent,
  PendingIntent.FLAG_UPDATE_CURRENT                    or
PendingIntent.FLAG_IMMUTABLE
 )

 val triggerTime = Calendar.getInstance().apply {
  add(Calendar.SECOND, 5)
```

```
 }.timeInMillis

 alarmManager.setExact(AlarmManager.RTC_WAKEUP,
triggerTime, pendingIntent)
 Toast.makeText(this,   "Alarm   Set   for   5   seconds",
Toast.LENGTH_SHORT).show()
 }
}
```

Langkah 1.4: Membuat *Activity* baru dengan nama NotificationService dan buat kode seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.Service
import android.content.Context
import android.content.Intent
import android.os. Build
import android.os.IBinder
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat

class NotificationService : Service() {

 override fun onCreate() {
 super.onCreate()
 // Create the notification channel
 createNotificationChannel()
 }

 override fun onStartCommand(intent: Intent?, flags: Int,
startId: Int): Int {
 sendNotification()
 return START_NOT_STICKY
 }

 private fun sendNotification() {
 val notificationId = 2
 val channelId = "service_channel"

 val   notification   =   NotificationCompat.   Builder(this,
channelId)
   .setSmallIcon(android.R.drawable.ic_dialog_info)
   .setContentTitle("Service Notification")
   .setContentText("This   is   a   notification   from   a
Service!")
   .setPriority(NotificationCompat.PRIORITY_HIGH)
```

```
  . Build()

 val                    notificationManager                    =
NotificationManagerCompat.from(this)
 notificationManager.notify(notificationId, notification)
 }

 private fun createNotificationChannel() {
 // Only create the channel on Android 8.0 (Oreo) and
higher
 if ( Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
  val channelId = "service_channel"
  val channelName = "Service Notifications"
  val channelDescription = "Notifications for background
service"
  val importance = NotificationManager.IMPORTANCE_HIGH

  val      channel      =      NotificationChannel(channelId,
channelName, importance).apply {
  description = channelDescription
  }

  // Register the channel with the system
  val                    notificationManager                    =
getSystemService(NotificationManager::class.java)
  notificationManager?.createNotificationChannel(channel)
 }
 }

 override fun onBind(intent: Intent?): IBinder? {
 return null
 }
}
```

Langkah 1.5: Membuat *Activity* baru dengan nama AlarmReceiver dan buat kode seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

import android.app.NotificationChannel
import android.app.NotificationManager
import android.content.BroadcastReceiver
import android.content.Context
import android.content.Intent
import android.os. Build
import androidx.core.app.NotificationCompat

class AlarmReceiver : BroadcastReceiver() {
```

```
 override fun onReceive(context: Context, intent: Intent)
{
 val                  notificationManager                  =
context.getSystemService(Context.NOTIFICATION_SERVICE)   as
NotificationManager

 // Buat NotificationChannel jika Android versi Oreo atau
lebih tinggi
 if ( Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
  val channel = NotificationChannel(
  "ALARM_CHANNEL_ID",
  "Alarm Notification",
  NotificationManager.IMPORTANCE_HIGH
  ).apply {
  description = "Notification channel for alarm"
  }
  notificationManager.createNotificationChannel(channel)
 }

 // Buat notifikasi
 val  notification  =  NotificationCompat.  Builder(context,
"ALARM_CHANNEL_ID")
  .setSmallIcon(R.drawable.ic_alarm) // Ganti dengan ikon
yang sesuai
  .setContentTitle("Alarm Triggered!")
  .setContentText("This is your alarm notification.")
  .setPriority(NotificationCompat.PRIORITY_HIGH)
  .setAutoCancel(true)
  . Build()

 // Tampilkan notifikasi
 notificationManager.notify(1001, notification)
 }
}
```

Langkah 1.6: Perbarui manifest dengan menambahkan kode berikut:

```
<uses-Permission
android:name="android.Permission.POST_NOTIFICATIONS" />
<uses-Permission
android:name="android.Permission.SCHEDULE_EXACT_ALARM" />

<service
android:name=".modul9notification.NotificationService" />
<receiver  android:name=".modul9notification.AlarmReceiver"
/>
```

2. Menambahkan Komponen UI pada Layout

Langkah 2.1: Perbarui *Activity*_main.xml dengan menambahkan kode berikut:Ketik konten Sub Bab disini

```
<Button
 android:id="@+id/button7"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Topik 7 - Notifikasi" />
```

Langkah 2.2: Isikan kode xml ke *Activity*_notification seperti berikut:

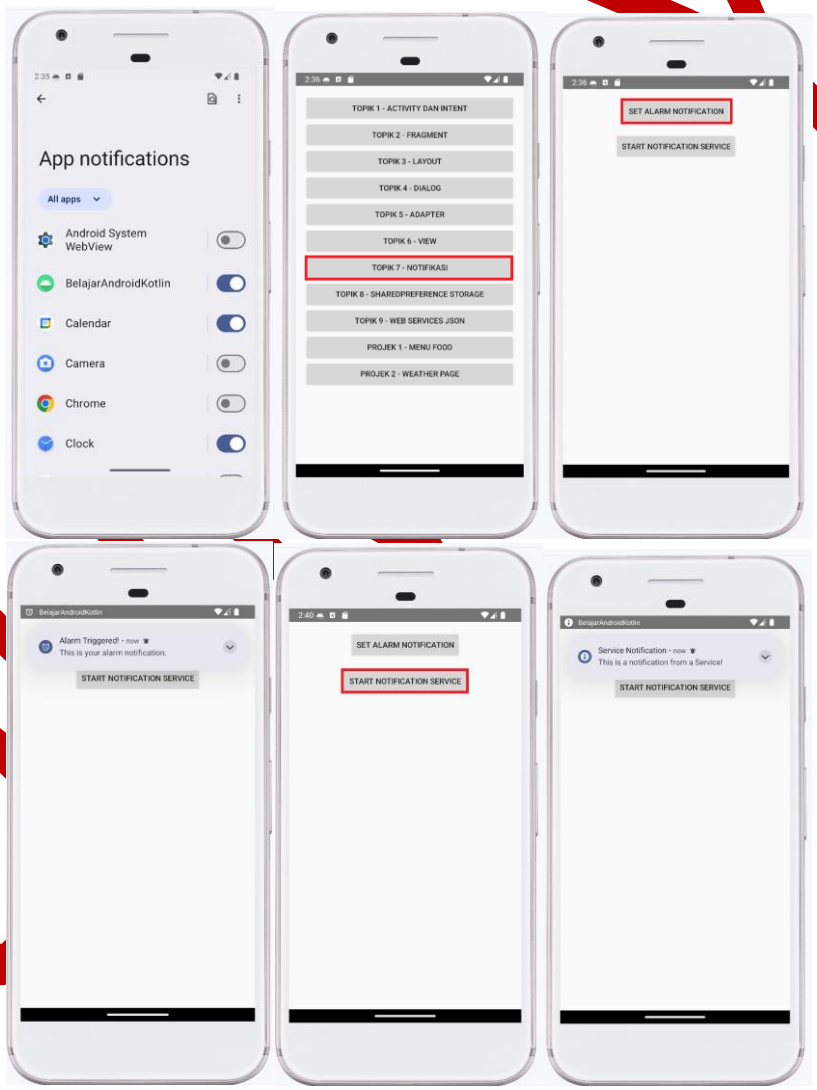
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android
"
 xmlns:app="
http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".Topik7.NotificationActvitiy">

 <Button
 android:id="@+id/buttonSetAlarm"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Set Alarm Notification"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginTop="16dp" />

 <Button
 android:id="@+id/buttonStartService"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Start Notification Service"
 app:layout_constraintTop_toBottomOf="@id/buttonSetAlarm"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 android:layout_marginTop="16dp" />

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

3. Hasil Pengujian

Langkah 3.1: Enable akses *Permission* untuk aplikasi pada android device.

Agar dapat menjalankan notifikasi, *Permission* notifications pada device android harus di enable terlebih dahulu. Silahkan akses menu setting pada device dan masuk app notifications.

Setelah enable run kembali aplikasi. Maka tampilan akan muncul seperti berikut:



Gambar 14. Hasil Pengujian Notifikasi Android

E. Tugas

1. **Modifikasi** : Setelah menyelesaikan langkah-langkah praktikum di atas lanjutkan modifikasi membuat notifikasi yang dapat tampil saat aplikasi sleep.
2. **Eksplorasi :** Penggunaan notifikasi dengan badge yang dapat menampilkan angka counting otomatis dan muncul pada icon launcher aplikasi.
3. **Laporan:** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 10
SHAREDPREFERENCES, INTERNAL STORAGE, DAN EXTERNAL STORAGE

Dalam pengembangan aplikasi Android, pengelolaan data adalah aspek penting yang mempengaruhi kinerja, keamanan, dan pengalaman pengguna. Android menyediakan beberapa metode penyimpanan data, yaitu SharedPreferences, Internal Storage, dan External Storage, yang masing-masing memiliki kelebihan dan kekurangan sesuai dengan jenis data yang disimpan dan kebutuhan aplikasi.

SharedPreferences cocok untuk menyimpan data sederhana dalam format pasangan kunci-nilai, seperti preferensi pengguna dan pengaturan aplikasi. Internal Storage memberikan penyimpanan yang lebih aman untuk data aplikasi yang lebih sensitif dan hanya bisa diakses oleh aplikasi itu sendiri. Sementara External Storage menyediakan kapasitas lebih besar dan fleksibilitas, namun perlu perhatian ekstra terkait dengan keamanan dan kontrol akses data.

Bab ini akan membahas cara menggunakan ketiga metode penyimpanan ini, kelebihan, kekurangannya, serta praktik terbaik dalam memilih metode penyimpanan yang tepat berdasarkan kebutuhan aplikasi, dengan fokus pada efisiensi dan keamanan data.

A. SharedPreferences

SharedPreferences adalah salah satu mekanisme penyimpanan data sederhana dalam Android yang memungkinkan aplikasi untuk menyimpan informasi dalam format pasangan key-value. Data yang disimpan dalam SharedPreferences bersifat persisten dan akan tetap tersedia meskipun aplikasi dimatikan atau perangkat di-restart, hingga data tersebut dihapus baik oleh aplikasi itu sendiri atau oleh pengguna. SharedPreferences menyimpan data dalam file XML yang ditempatkan di direktori internal aplikasi **(data/data/<nama_paket_aplikasi>/shared_prefs)** yang tidak dapat diakses oleh aplikasi lain secara default, menjadikannya aman dari intervensi aplikasi lain.

SharedPreferences umumnya digunakan untuk menyimpan data kecil dan sederhana yang sering digunakan dalam pengaturan aplikasi

DUMMY

atau preferensi pengguna, seperti status login, bahasa aplikasi, atau preferensi tampilan (misalnya mode gelap atau terang).

1. Cara Kerja SharedPreferences

   SharedPreferences bekerja melalui penggunaan pasangan key-value, di mana setiap data memiliki pasangan unik berupa key yang berfungsi sebagai pengenal dari data tersebut. Aplikasi akan menyimpan informasi ini dalam file XML di direktori internal yang aman dan secara otomatis dikelola oleh sistem. Ketika aplikasi membutuhkan data, file XML ini akan diakses untuk mendapatkan nilai yang terkait dengan key yang sesuai.

   Secara sederhana, SharedPreferences menghindarkan pengembang dari kebutuhan untuk mengimplementasikan basis data yang lebih kompleks seperti SQLite, terutama ketika yang disimpan adalah data dalam jumlah kecil dan sederhana. Operasi penyimpanan dilakukan melalui metode put yang bersifat asinkron untuk memastikan performa aplikasi tetap optimal. Data kemudian dapat diambil dengan metode get yang memungkinkan aplikasi membaca nilai yang sebelumnya telah disimpan.

2. Fitur-fitur SharedPreferences
   a. Persistent: Data yang disimpan dalam SharedPreferences tetap ada walaupun aplikasi ditutup atau perangkat di-restart, kecuali jika data tersebut dihapus oleh aplikasi atau pengguna.
   b. Private secara Default: File yang menyimpan data SharedPreferences tidak dapat diakses oleh aplikasi lain, sehingga data bersifat aman.
   c. Mudah Digunakan: Akses terhadap data dalam SharedPreferences dilakukan melalui metode put dan get, menjadikan proses penyimpanan dan pengambilan data lebih cepat dan praktis.
3. Kelebihan SharedPreferences
   a. Kemudahan Implementasi: SharedPreferences sangat mudah digunakan dan diimplementasikan, bahkan oleh pengembang pemula. Cukup dengan menentukan pasangan key-value, pengembang dapat menyimpan dan mengambil data tanpa memerlukan basis data atau struktur penyimpanan yang kompleks.
   b. Efisiensi untuk Data Sederhana: SharedPreferences ideal untuk penyimpanan data yang sederhana dan sering diakses, seperti pengaturan pengguna atau preferensi tampilan.
   c. Keamanan Data: Karena tersimpan di direktori internal aplikasi, data yang disimpan dalam SharedPreferences hanya dapat diakses

DUMMY

oleh aplikasi yang bersangkutan, menjadikannya aman dari akses aplikasi pihak ketiga.

4. Kekurangan SharedPreferences
   a. Terbatas pada Data Sederhana: SharedPreferences dirancang hanya untuk menyimpan data sederhana. Aplikasi yang membutuhkan penyimpanan data dalam jumlah besar atau data yang kompleks (seperti daftar objek atau struktur data yang berlapis) akan membutuhkan solusi penyimpanan lain seperti SQLite atau Room Database.
   b. Tidak Mendukung Tipe Data Kompleks: SharedPreferences hanya mendukung tipe data primitif seperti int, boolean, float, long, dan String. Untuk menyimpan tipe data yang lebih kompleks, pengembang perlu melakukan serialisasi data menjadi bentuk string atau menggunakan format JSON.
   c. Kapasitas Penyimpanan Terbatas: SharedPreferences tidak cocok untuk menyimpan data dalam ukuran besar, karena keterbatasan kapasitas penyimpanan dan performa yang dapat terpengaruh jika file menjadi terlalu besar.
5. Implementasi SharedPreferences

   Berikut ini adalah langkah-langkah implementasi SharedPreferences dalam aplikasi Android, dimulai dari membuat instance hingga menyimpan dan mengambil data:

   a. Membuat atau Mendapatkan Instance dari SharedPreferences

      Untuk menggunakan SharedPreferences, pertama-tama kita harus membuat atau mendapatkan instance SharedPreferences dengan nama khusus yang spesifik untuk aplikasi. Nama ini berguna untuk mengelompokkan data yang disimpan dalam file XML tertentu.SharedPreferences Ketik konten Sub Bab disini

      ```
      val sharedPreferences =
      getSharedPreferences("MyAppPreferences",
      Context.MODE_PRIVATE)
      ```

      Context.MODE_PRIVATE menjadikan file tersebut private, yang berarti hanya dapat diakses oleh aplikasi yang membuatnya.

   b. Menyimpan Data Menggunakan Metode **put**

      Untuk menyimpan data dalam SharedPreferences, kita menggunakan Editor yang berfungsi untuk menulis pasangan key-value ke dalam file SharedPreferences. apply() digunakan untuk

DUMMY

menyimpan data secara asinkron, mengoptimalkan performa aplikasi karena tidak akan memblokir proses utama.

```
val editor = sharedPreferences.edit()
editor.putString("username", "User123") // Menyimpan data String
editor.putBoolean("isLoggedIn", true) // Menyimpan data Boolean
editor.apply() // Menyimpan perubahan secara asynchronous
```

c. Mengambil Data Menggunakan Metode **get**

Untuk mengakses data yang disimpan, kita menggunakan metode get yang mengembalikan nilai dari pasangan key-value yang diinginkan. Kita juga dapat menetapkan nilai default jika key yang dimaksud belum ada dalam SharedPreferences.

```
val username = sharedPreferences.getString("username", "") // Mengambil data String
val isLoggedIn = sharedPreferences.getBoolean("isLoggedIn", false) // Mengambil data Boolean
```

d. Menghapus Data dari SharedPreferences

SharedPreferences juga menyediakan cara untuk menghapus data jika tidak lagi dibutuhkan, dengan menggunakan remove atau clear pada objek Editor.

```
editor.remove("username") // Menghapus data dengan key "username"
editor.clear() // Menghapus semua data di SharedPreferences
editor.apply() // Menyimpan perubahan
```

B. Internal Storage

Internal Storage adalah salah satu metode penyimpanan data di Android yang bersifat pribadi dan eksklusif untuk setiap aplikasi. Data yang disimpan menggunakan Internal Storage hanya dapat diakses oleh aplikasi yang menyimpan data tersebut. Aplikasi lain tidak dapat mengakses data ini kecuali ada izin eksplisit yang diberikan oleh aplikasi yang bersangkutan. Karena keamanan dan privasi yang terjaga dengan baik, Internal Storage menjadi pilihan yang tepat untuk menyimpan data sensitif atau berkas yang bersifat pribadi, seperti informasi akun pengguna, preferensi khusus, atau berkas lain yang penting.

DUMMY

Berbeda dengan penyimpanan eksternal, data di Internal Storage tidak dapat dilihat atau diakses oleh aplikasi lain, sehingga pengguna dapat merasa lebih aman dan terlindungi. Data yang disimpan di Internal Storage bersifat persisten, artinya data tersebut tetap ada meskipun aplikasi di-restart, hingga data dihapus secara manual oleh pengguna atau saat aplikasi dihapus dari perangkat.

1. Cara Kerja Internal Storage

   Internal Storage bekerja dengan menyediakan ruang penyimpanan khusus yang dikelola di dalam direktori aplikasi di perangkat. Android menyimpan data di direktori data/data/<nama_paket_aplikasi>/files, yang hanya bisa diakses oleh aplikasi tersebut. Setiap kali aplikasi menyimpan data, sistem akan menciptakan file di dalam direktori tersebut, dan file tersebut bersifat private secara default. Oleh karena itu, file ini tidak dapat diakses oleh aplikasi lain, yang menjadikannya aman untuk menyimpan data sensitif.

   Untuk menyimpan data di Internal Storage, pengembang Android menggunakan metode openFileOutput, yang memungkinkan aplikasi untuk menulis data ke file. Sebaliknya, data dapat diambil atau dibaca kembali dengan metode openFileInput. Data yang disimpan dalam Internal Storage ini akan tetap ada hingga pengguna menghapusnya secara manual atau ketika aplikasi dihapus.

2. Fitur-fitur Internal Storage
   a. Private Secara Default: Data yang disimpan di Internal Storage hanya dapat diakses oleh aplikasi yang menyimpannya, sehingga lebih aman dari intervensi pihak ketiga.
   b. Keamanan Tinggi: Dibandingkan metode penyimpanan lainnya, Internal Storage menawarkan tingkat keamanan yang lebih tinggi karena data tidak dapat diakses dari aplikasi lain.
   c. Cocok untuk Data Sensitif: Internal Storage adalah pilihan tepat untuk menyimpan data yang memerlukan privasi dan keamanan, seperti data pengguna yang sensitif atau file pribadi.
3. Kelebihan Internal Storage
   a. Keamanan Tinggi: Internal Storage sangat aman karena data yang disimpan tidak bisa diakses oleh aplikasi lain tanpa izin khusus, sehingga data pengguna terlindungi dari akses tidak sah.
   b. Pilihan Terbaik untuk Data Sensitif: Karena aksesnya yang terbatas dan aman, Internal Storage sangat ideal untuk menyimpan informasi pribadi atau data yang memerlukan perlindungan lebih.

DUMMY

c. Data Persistent: Data yang disimpan di Internal Storage bersifat persisten, sehingga tetap ada dan tidak terhapus meskipun aplikasi dimatikan atau perangkat di-restart, kecuali aplikasi atau pengguna menghapus data tersebut.
d. Tidak Terpengaruh oleh Akses Kartu SD: Data yang disimpan di Internal Storage tidak akan terpengaruh oleh penghapusan atau pencabutan kartu SD karena data berada di penyimpanan internal perangkat.

4. Kekurangan Internal Storage
a. Kapasitas Penyimpanan Terbatas: Kapasitas Internal Storage sangat bergantung pada kapasitas penyimpanan internal perangkat. Untuk perangkat dengan kapasitas penyimpanan kecil, hal ini bisa menjadi batasan signifikan, terutama jika aplikasi memerlukan penyimpanan data yang besar.
b. Tidak Efisien untuk File Berukuran Besar: Internal Storage tidak cocok untuk menyimpan data yang berukuran besar, seperti file media atau berkas yang memakan ruang penyimpanan yang cukup banyak.
c. Meningkatkan Ukuran Total Aplikasi: Data yang disimpan dalam Internal Storage akan menambah ukuran total aplikasi pada perangkat. Semakin banyak data yang disimpan, semakin besar pula ukuran aplikasi.

5. Implementasi Internal Storage
Untuk menyimpan dan mengambil data dari Internal Storage, Android menyediakan metode openFileOutput untuk menulis data ke file dan openFileInput untuk membaca data dari file yang disimpan. Penggunaan metode ini sangat mudah, dan dapat disesuaikan dengan kebutuhan aplikasi.
Berikut ini adalah langkah-langkah implementasi Internal Storage dalam aplikasi Android:
a. Membuat atau Menyimpan Data ke File di Internal Storage
Untuk menyimpan data, pengembang menggunakan metode openFileOutput. Metode ini menciptakan atau membuka file dengan nama yang diberikan di direktori aplikasi, dan menulis data yang diinginkan ke file tersebut.

```
// Menyimpan data ke file di Internal Storage
val fileName = "dataUser.txt"   // Nama file
val fileContent = "User: John Doe, ID: 12345" // Isi file
openFileOutput(fileName, Context.MODE_PRIVATE).use {
```

DUMMY

```
 it.write(fileContent.toByteArray())    // Menulis data
ke dalam file
}
```

Dalam contoh di atas:

1) fileName adalah nama file yang akan disimpan di Internal Storage.
2) fileContent adalah konten atau isi dari file tersebut.
3) Context.MODE_PRIVATE memastikan file hanya dapat diakses oleh aplikasi yang membuatnya.

b. Membaca Data dari File di Internal Storage

Data yang telah disimpan dapat dibaca kembali dengan menggunakan metode openFileInput, yang membuka file sesuai dengan nama yang diberikan dan membaca isi file tersebut.

```
// Membaca data dari file di Internal Storage
openFileInput(fileName).bufferedReader().useLines      {
lines ->
 lines.forEach {
 println(it)       // Mencetak setiap baris konten dari
file
 }
}
```

Dalam contoh ini, openFileInput membuka file dataUser.txt yang sudah tersimpan di Internal Storage. bufferedReader().useLines digunakan untuk membaca setiap baris dalam file.

c. Menghapus File dari Internal Storage

Jika data yang disimpan tidak lagi diperlukan, file yang menyimpan data tersebut dapat dihapus menggunakan metode deleteFile.

C. External Storage

External Storage adalah jenis penyimpanan data pada perangkat Android yang dapat diakses oleh aplikasi lain maupun oleh pengguna secara langsung. External Storage meliputi penyimpanan eksternal yang dapat dilepas, seperti kartu SD, maupun penyimpanan eksternal yang tidak dapat dilepas, yang sering kali terintegrasi dalam perangkat itu sendiri. Metode penyimpanan ini sangat cocok untuk data berukuran besar, seperti foto, video, musik, atau dokumen, yang mungkin perlu diakses oleh beberapa aplikasi atau langsung oleh pengguna melalui pengelola berkas (file manager).

DUMMY

Data yang disimpan pada External Storage dapat diakses secara luas oleh aplikasi lain di perangkat yang sama. Namun, data di External Storage juga lebih rentan dihapus baik oleh pengguna atau oleh sistem ketika ruang penyimpanan terbatas. Untuk itu, External Storage umumnya digunakan untuk file-file publik atau konten multimedia yang tidak bersifat sensitif. Aplikasi yang menyimpan data sensitif atau pribadi pengguna disarankan menggunakan Internal Storage yang lebih aman.

1. Cara Kerja External Storage

   Sistem operasi Android membedakan antara public dan private External Storage:

   a. Public External Storage

      Jenis penyimpanan ini memungkinkan aplikasi untuk menyimpan data secara publik di ruang yang dapat diakses oleh aplikasi lain maupun pengguna melalui file manager. Data yang disimpan di Public External Storage dapat tetap ada meskipun aplikasi dihapus, tetapi karena sifatnya yang terbuka, data ini juga rentan terhadap akses atau modifikasi oleh aplikasi lain atau oleh pengguna itu sendiri.

   b. Private External Storage

      Private External Storage adalah ruang di External Storage yang hanya dapat diakses oleh aplikasi yang menyimpan data tersebut. Data yang disimpan pada Private External Storage hanya dapat diakses selama aplikasi masih terpasang. Ketika aplikasi dihapus, data di Private External Storage juga akan dihapus, sehingga ruang ini lebih cocok untuk data aplikasi yang bersifat sementara dan tidak terlalu sensitif.

      Saat menyimpan data di External Storage, aplikasi perlu mengecek ketersediaan penyimpanan dan meminta izin yang relevan dari pengguna, terutama sejak Android 6.0 (API level 23) yang memperkenalkan runtime *Permission*s. Dengan langkah ini, aplikasi hanya akan memiliki akses ke penyimpanan eksternal jika pengguna secara eksplisit memberikan izin.

2. Fitur External Storage

   a. Ruang Penyimpanan Lebih Besar: External Storage umumnya menawarkan kapasitas penyimpanan yang lebih besar dibandingkan Internal Storage, terutama jika perangkat mendukung penggunaan kartu SD yang berkapasitas tinggi.

DUMMY

b. Akses oleh Aplikasi Lain: Data yang disimpan di External Storage dapat diakses oleh aplikasi lain, memungkinkan interoperabilitas antar aplikasi yang membutuhkan data yang sama.
c. Cocok untuk File Berukuran Besar: External Storage sangat ideal untuk menyimpan file besar seperti video, gambar, atau dokumen yang membutuhkan kapasitas penyimpanan besar.

3. Kelebihan External Storage
   a. Ruang Penyimpanan Luas: Kapasitas yang lebih besar di External Storage membuatnya cocok untuk menyimpan file multimedia dan dokumen besar tanpa membebani Internal Storage.
   b. Aksesibilitas untuk Aplikasi Lain: Karena dapat diakses oleh aplikasi lain, file yang disimpan di External Storage bisa diolah oleh berbagai aplikasi yang kompatibel atau bahkan langsung diakses pengguna.
   c. Ideal untuk File yang Dibagikan: File yang perlu dibagikan antar aplikasi atau diakses pengguna secara langsung melalui pengelola berkas sebaiknya disimpan di External Storage agar lebih mudah diakses dan dikelola.
4. Kekurangan External Storage
   a. Keamanan Lebih Rendah: Data yang disimpan di External Storage lebih mudah diakses oleh aplikasi lain dan pengguna, sehingga kurang cocok untuk data yang bersifat pribadi atau sensitif.
   b. Data Tidak Persistent: Data di External Storage dapat dihapus jika pengguna secara manual menghapusnya atau jika kartu SD dilepas dari perangkat, membuat penyimpanan ini kurang andal untuk data jangka panjang.
   c. Izin yang Lebih Rumit: Untuk mengakses External Storage, aplikasi harus meminta izin khusus dari pengguna. Mulai Android 6.0, aplikasi harus meminta izin READ_EXTERNAL_STORAGE dan/atau WRITE_EXTERNAL_STORAGE untuk mengakses atau menyimpan data di penyimpanan eksternal.
5. Implementasi External Storage

   Dalam implementasi, ada beberapa langkah yang perlu diperhatikan untuk menggunakan External Storage:
   a. Periksa Status Penyimpanan Eksternal

      Sebelum menulis data ke External Storage, pastikan penyimpanan eksternal dalam status mounted atau siap untuk diakses. Android menyediakan metode Environment.getExternalStorageState() untuk memeriksa status ini.

DUMMY

b. Izin Akses Penyimpanan
Aplikasi harus memastikan izin yang diperlukan sudah diberikan oleh pengguna. Untuk perangkat dengan Android 6.0 atau lebih baru, izin akses penyimpanan eksternal perlu diminta secara eksplisit melalui runtime *Permission*s.

c. Menyimpan dan Membaca Data dari External Storage
Gunakan direktori yang sesuai dengan kebutuhan aplikasi. Contohnya, untuk dokumen dapat menggunakan Environment.DIRECTORY_DOCUMENTS, sedangkan untuk gambar atau media lain dapat menggunakan Environment.DIRECTORY_PICTURES atau Environment.DIRECTORY_MUSIC.
Contoh implementasi untuk menyimpan data ke file di External Storage:

```
import android.content.Context
import android.os.Environment
import java.io.File

// Periksa apakah External Storage tersedia untuk ditulis
if (Environment.MEDIA_MOUNTED == Environment.getExternalStorageState()) {
 // Tentukan direktori dan nama file
 val file = File(context.getExternalFilesDir(Environment.DIRECTORY_DOCUMENTS), "sample.txt")

 // Tulis data ke file di External Storage
 file.writeText("Data disimpan di External Storage")
   }
```

Pada contoh di atas, file "sample.txt" disimpan di direktori dokumen milik aplikasi dalam External Storage. Direktori ini bersifat private, sehingga hanya dapat diakses oleh aplikasi itu sendiri, dan akan dihapus secara otomatis jika aplikasi dihapus dari perangkat.
Dengan mengikuti prinsip dan praktik terbaik dalam penggunaan External Storage, aplikasi dapat memberikan fleksibilitas yang lebih besar dalam penyimpanan data tanpa mengorbankan keamanan dan konsistensi data yang diperlukan dalam sistem operasi Android.

D. Implementasi SharedPreferences

Langkah - Langkah implementasi pada aplikasi

1. Membuat *Activity*

   Langkah 1.1: Perbarui Main *Activity* sebelumnya dengan menambahkan kode berikut:

```
package com.kurniawan.belajarandroidkotlin.Topik8

import android.content.Context
import android.content.SharedPreferences
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import androidx.appcompat.app.AppCompat Activity
import com.kurniawan.belajarandroidkotlin.R

class Shared Activity : AppCompat Activity() {

 private lateinit var sharedPreferences:
SharedPreferences
 private lateinit var editor: SharedPreferences.Editor
 private lateinit var editTextName: EditText
 private lateinit var btnSave: Button
 private lateinit var btnLoad: Button
 private lateinit var tvResult: TextView

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout. Activity_shared)


 sharedPreferences = getSharedPreferences("UserPrefs",
Context.MODE_PRIVATE)
 editor = sharedPreferences.edit()

 editTextName = findViewById(R.id.editTextName)
 btnSave = findViewById(R.id.btnSave)
 btnLoad = findViewById(R.id.btnLoad)
 tvResult = findViewById(R.id.tvResult)

 btnSave.setOnClickListener {
  val name = editTextName.text.toString()
  if (name.isNotEmpty()) {
  editor.putString("user_name", name)
  editor.apply() // Apply data yang disimpan
  }
 }
```

```
 btnLoad.setOnClickListener {
  val name = sharedPreferences.getString("user_name", "No
name saved")
  tvResult.text = name
 }
 }
}
```

2. Menambahkan Komponen UI pada Layout

   Langkah 2.1: Perbarui *Activity*_main.xml dengan menambahkan kode berikut:

```
<Button
 android:id="@+id/button8"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Topik 8 - Sharedpreference" />
```

   Langkah 2.2: Isikan kode xml ke *Activity*_shared seperti berikut:

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

 <EditText
 android:id="@+id/editTextName"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:hint="Enter your name"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

 <Button
 android:id="@+id/btnSave"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Save Name"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@id/editTextName"
 android:layout_marginTop="16dp"/>
```

```
    <Button
    android:id="@+id/btnLoad"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Load Name"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/btnSave"
    android:layout_marginTop="16dp"/>

    <TextView
    android:id="@+id/tvResult"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Name will appear here"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/btnLoad"
    android:layout_marginTop="16dp" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Pengujian Aplikasi
   Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.
   Langkah 3.1: Klik Run pada toolbar Android Studio atau tekan Shift + F10 untuk memulai proses kompilasi dan menjalankan aplikasi.
   Langkah 3.2: Pada jendela Select Deployment Target, pilih perangkat yang akan digunakan untuk menguji aplikasi:
   a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
   b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode USB Debugging aktif.
4. Hasil PengujianKetik konten Sub Bab disini

DUMMY

Gambar 15. Hasil Pengujian Sharedpreferences, Internal Storage, dan External Storage

E. Tugas

1. **Eksplorasi :** Buatlah aplikasi sederhana yang menyimpan dan menampilkan preferensi pengguna, seperti nama dan status login menggunakan SharedPreferences.
2. **Laporan :** Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.Ketik konten Sub Bab disini

DUMMY

BAB 11
WEB SERVICE JSON OBJECT & JSON ARRAY HTTP REQUEST DENGAN RETROFIT

Dalam pengembangan aplikasi Android yang modern, salah satu fitur penting yang sering digunakan adalah kemampuan aplikasi untuk berkomunikasi dengan server. Komunikasi ini memungkinkan aplikasi untuk mengambil data secara real-time dari server dan menampilkan informasi yang up-to-date kepada pengguna. Misalnya, saat kita membuka aplikasi media sosial, aplikasi akan mengambil data terbaru seperti postingan, komentar, atau pesan langsung dari server menggunakan koneksi internet.

Untuk mengimplementasikan komunikasi antara aplikasi Android dan server, kita menggunakan Web Service. Web Service menyediakan antarmuka yang memungkinkan aplikasi Android mengirim dan menerima data melalui jaringan menggunakan protokol HTTP. Data yang dikirimkan antara server dan aplikasi biasanya diformat dalam bentuk JSON (JavaScript Object Notation), karena format ini ringan dan mudah dibaca oleh manusia serta diproses oleh mesin.

Namun, membuat dan mengelola HTTP request secara manual dapat menjadi tugas yang rumit, terutama saat mengatur proses parsing data JSON, error handling, dan pengelolaan request asynchronous. Untuk menyederhanakan proses ini, library seperti Retrofit digunakan dalam pengembangan aplikasi Android. Retrofit adalah library yang sangat populer untuk mengelola HTTP request dan mempermudah pengembang dalam mengakses API RESTful. Dengan Retrofit, pengembang dapat dengan mudah melakukan HTTP request seperti GET, POST, PUT, dan DELETE tanpa perlu menulis kode yang kompleks.

DUMMY

A. JSON (JavaScript Object Notation) J

JSON (JavaScript Object Notation) adalah format pertukaran data berbasis teks yang digunakan secara luas dalam pengembangan aplikasi berbasis web dan mobile. JSON didesain untuk mudah dibaca oleh manusia serta mudah diparsing atau dikonversi oleh mesin. JSON sering digunakan untuk mengirim data antar server dan klien dalam aplikasi berbasis RESTful API.Ketik konten Sub Bab disini.

Jenis Struktur JSON

1. JSON Object

   JSON Object adalah struktur data yang menggunakan pasangan key-value, mirip dengan konsep dictionary pada Python atau objek pada JavaScript. JSON Object biasanya digunakan untuk mendeskripsikan satu entitas atau objek. Contoh JSON Object:

   ```
   {
    "id": 1,
    "title": "Belajar Retrofit",
    "author": "Budi",
    "year": 2024
   }
   ```

2. JSON Array

   JSON Array adalah struktur data yang terdiri dari daftar elemen, yang bisa berupa nilai primitif (number, string, boolean) atau objek. JSON Array biasanya digunakan untuk menyimpan kumpulan data atau daftar entitas. Contoh JSON Array:

   ```
   [
    {"id": 1, "title": "Android Basics"},
    {"id": 2, "title": "Advanced Kotlin"},
    {"id": 3, "title": "Retrofit in Android"}
   ]
   ```

   Pada contoh di atas, JSON Array berisi tiga objek dengan struktur yang serupa. Masing-masing objek merepresentasikan sebuah buku dengan key id dan title.

Kelebihan JSON:

1. JSON lebih sederhana dan lebih ringkas dibandingkan dengan format lain seperti XML.
2. JSON didukung oleh hampir semua bahasa pemrograman, termasuk Java, Kotlin, Python, JavaScript, dan banyak lagi.
3. JSON adalah format standar untuk pertukaran data dalam API RESTful, membuatnya sangat kompatibel dengan berbagai layanan web.

B. Web Service

Web Service adalah layanan yang memungkinkan pertukaran data antara aplikasi dan server melalui jaringan internet. Web Service

DUMMY

menyediakan antarmuka untuk aplikasi agar dapat berkomunikasi dengan server secara efektif, memungkinkan pengiriman data dari klien (aplikasi) ke server maupun penerimaan data dari server ke klien. Dalam dunia pengembangan aplikasi Android, Web Service sering kali diimplementasikan menggunakan API (Application Programming Interface) yang memanfaatkan protokol HTTP/HTTPS untuk mengirim dan menerima data.

Web Service bekerja dengan menggunakan prinsip client-server, di mana aplikasi Android bertindak sebagai klien yang mengirimkan permintaan (HTTP request) ke server, dan server kemudian memberikan tanggapan (HTTP response) kepada klien. Dengan cara ini, aplikasi dapat memperoleh data secara dinamis, seperti daftar produk, informasi cuaca, berita terkini, atau data profil pengguna. Proses ini melibatkan penggunaan metode HTTP yang berbeda tergantung pada jenis operasi yang ingin dilakukan.

Berikut adalah beberapa metode HTTP yang umum digunakan dalam Web Service:

1. **GET:**
   Metode GET digunakan untuk mengambil data dari server. Permintaan ini tidak akan mengubah data apa pun di server. Biasanya, metode GET digunakan untuk mengambil informasi yang sudah ada, seperti daftar artikel, produk, atau profil pengguna.
   Contoh: saat pengguna membuka aplikasi berita dan ingin melihat daftar artikel terbaru, aplikasi mengirim permintaan GET ke server untuk mengambil daftar artikel tersebut.
   Contoh kode:
   GET /api/articles
2. **POST:**
   Metode POST digunakan untuk mengirim data baru ke server untuk disimpan. Metode ini biasanya digunakan saat pengguna mengirimkan formulir atau data input ke server, seperti saat mendaftar akun baru atau mengirim pesan.
   Contoh: Ketika pengguna mengisi formulir pendaftaran pada aplikasi dan menekan tombol "Daftar", aplikasi mengirim permintaan POST ke server dengan data pengguna.
   Contoh kode:
   POST /api/users/register
   Content-Type: application/json

DUMMY

```
{
 "username": "Budi",
 "email": "budi@example.com",
 "password": "password123"
}
```

3. **PUT:**
   Metode PUT digunakan untuk memperbarui atau mengubah data yang sudah ada di server. Data yang dikirim melalui permintaan PUT akan menggantikan data yang ada di server dengan data baru yang dikirim oleh klien.
   Contoh: Ketika pengguna mengubah informasi profilnya, seperti mengupdate nama atau email, aplikasi mengirim permintaan PUT ke server dengan data yang baru.
   Contoh Kode:

```
PUT /api/users/1
Content-Type: application/json

{
 "username": "Budi_updated",
 "email": "Budi_new@example.com"
}
```

4. **DELETE:**
   Metode DELETE digunakan untuk menghapus data yang ada di server. Permintaan ini digunakan ketika pengguna ingin menghapus sumber daya tertentu, seperti menghapus akun atau postingan.
   Contoh: Saat pengguna memilih untuk menghapus akunnya dari aplikasi, aplikasi mengirim permintaan DELETE ke server.
   Contoh Kode:

```
DELETE /api/users/1
```

C. Retrofit

Retrofit adalah library HTTP client yang dikembangkan oleh Square untuk memudahkan pengembang Android dalam mengakses API RESTful. Library ini sangat populer karena menyederhanakan proses pengelolaan HTTP request dan response. Retrofit dapat menangani berbagai format data, seperti JSON dan XML, dengan bantuan converter. Selain itu, Retrofit juga mendukung asynchronous request, yang

DUMMY

memungkinkan pengambilan data dari server dilakukan di latar belakang tanpa mengganggu antarmuka pengguna (UI) aplikasi.

Mengelola HTTP request secara manual bisa menjadi tugas yang rumit dan memakan waktu, terutama ketika melibatkan parsing data JSON, penanganan error, dan pengelolaan request secara asynchronous. Dengan Retrofit, pengembang dapat dengan mudah melakukan HTTP request, mengambil data dari server, dan mengelola response secara otomatis, sehingga membuat proses pengembangan aplikasi menjadi lebih efisien.

Keunggulan Retrofit

1. Sederhana dan Mudah Dipelajari
   Retrofit memiliki sintaks yang sederhana dan mudah dimengerti. Pengembang hanya perlu mendefinisikan endpoint API dalam interface, dan Retrofit akan menangani sebagian besar proses HTTP request tanpa perlu menulis kode HTTP request secara manual.
2. JSON Parsing Otomatis
   Retrofit dapat mengonversi data JSON yang diterima dari server menjadi objek Java atau Kotlin dengan menggunakan converter seperti Gson. Hal ini menghemat waktu pengembang karena tidak perlu menulis kode parsing JSON secara manual, yang biasanya memakan banyak waktu dan rentan terhadap kesalahan.
3. Support Asynchronous Request
   Retrofit mendukung request asynchronous, yang berarti proses pengambilan data dari server dapat dilakukan di latar belakang tanpa mengganggu antarmuka pengguna. Pengguna tetap dapat berinteraksi dengan aplikasi sementara data sedang diambil, yang meningkatkan pengalaman pengguna (UX).
4. Dukungan Error Handling
   Retrofit menyediakan mekanisme yang efektif untuk menangani kesalahan selama proses HTTP request. Pengembang dapat menangani berbagai jenis kesalahan seperti masalah koneksi, time-out, atau respons server yang tidak valid dengan cara yang terstruktur dan efisien. Dengan Retrofit, pengembang dapat memastikan aplikasi tetap berfungsi dengan baik meskipun terjadi kesalahan di sisi server atau koneksi.
5. Dukungan untuk Interceptor dan Logging
   Retrofit mendukung penggunaan Interceptor untuk memodifikasi HTTP request dan response sebelum dikirim atau diterima. Ini memungkinkan pengembang untuk menambahkan logika khusus,

DUMMY

seperti autentikasi atau modifikasi data sebelum pengiriman. Retrofit juga dapat diintegrasikan dengan OkHttp Logging Interceptor untuk mencatat detail request dan response, yang sangat berguna saat debugging dan memantau lalu lintas data aplikasi.

Mengelola HTTP Request dengan Retrofit

Mengelola HTTP request secara manual dapat menjadi pekerjaan yang memakan banyak waktu dan rawan kesalahan, terutama ketika harus menangani parsing data, pengelolaan error, dan request asynchronous. Dengan menggunakan Retrofit, proses ini menjadi lebih sederhana dan lebih efisien. Retrofit mengelola parsing data secara otomatis, memungkinkan pengembang fokus pada logika aplikasi, bukan pada detail komunikasi HTTP.

Dengan berbagai keunggulan yang ditawarkan, Retrofit menjadi salah satu pilihan terbaik untuk pengembangan aplikasi Android yang membutuhkan komunikasi dengan server menggunakan API. Retrofit mempermudah pengelolaan HTTP request dan response, menyediakan dukungan asynchronous untuk pengambilan data di latar belakang, serta memberikan mekanisme error handling yang baik dan efisien. Semua ini menjadikan Retrofit sebagai alat yang penting untuk mempercepat pengembangan aplikasi yang membutuhkan integrasi dengan layanan eksternal melalui HTTP.

D. Implementasi Web Service JSON Object & JSON Array HTTP Request dengan Retrofit

Langkah-Langkah Implementasi Aplikasi

1. Persiapan Proyek Android Studio

   Langkah 1.1: Buka project **BelajarAndroidKotlin** yang sudah dibuat sebelumnya.

   Langkah 1.2: Beri nama package **topik9** dengan cara klik kanan pada package **com.kurniawan.belajarandroidkotlin** lalu klik **New > package**

   Langkah 1.3: Buat package baru dan beri nama ***Activity*** dengan cara klik kanan pada package *topik9* lalu klik **New > package**

   Langkah 1.4: Buat package baru dan beri nama ***Adapter*** dengan cara klik kanan pada package *topik9* lalu klik **New > package**

   Langkah 1.5: Buat package baru dan beri nama **api** dengan cara klik kanan pada package *topik10* lalu klik **New > package**

DUMMY

Langkah 1.6: Buat package baru dan beri nama **model** dengan cara klik kanan pada package *topik10* lalu klik **New > package**

2. Membuat File-File Utama

Langkah 2.1: Buka file **Main *Activity*.kt** yang sudah dibuat sebelumnya.

Langkah 2.2: Tambahkan kode berikut untuk mendefinisikan tombol navigasi Web Service JSONKetik konten Sub Bab disini

```
val buttonJSON = findViewById<Button>(R.id.buttonJSON)
buttonJSON.setOnClickListener { it: View!
    val intent = Intent( packageContext: this, JsonActivity::class.java)
    startActivity(intent)
}
```

Langkah 2.3: Tambahkan kode pada layout ***Activity*_main.xml** di folder **res/layout/** untuk navigasi Main *Activity* tombol Web Service JSON:

```
<Button
 android:id="@+id/buttonJSON"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Topik 9 - Web Service JSON" />
```

3. Implementasi *Activity* pada package topik9

Langkah 3.1: Buat file **Json *Activity*.kt** di dalam package *Activity* dengan cara klik kanan pada package, pilih **New > *Activity* > Empty Views *Activity*.**

Langkah 3.2: Pada jendela pengaturan *Activity*, lakukan konfigurasi berikut:

a. Pada opsi ***Activity* Name**, ketik **Json *Activity*.**
b. Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
c. Pada bagian **Layout Name**, ketik ***Activity*_json.xml** sebagai nama layout.
d. Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.
e. Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.topik9**
f. Pada bagian Source Language, pastikan bahasa yang dipilih adalah Kotlin.

Langkah 3.3: Setelah semua pengaturan sesuai, klik **Finish** untuk membuat *Activity* baru beserta layout-nya.

Langkah 3.4: Tambahkan kode berikut pada **Json *Activity*.kt:**

DUMMY

JsonActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik9.activity

import android.content.Intent
import android.os.Bundle
import android.widget.Button
import androidx.appcompat.app.AppCompatActivity
import com.kurniawan.belajarandroidkotlin.R
import com.kurniawan.belajarandroidkotlin.topik9.activity.PostActivity

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

        val btnUser = findViewById<Button>(R.id.btnUser)
        val btnPost = findViewById<Button>(R.id.btnPost)

        btnUser.setOnClickListener { it: View!
            Intent( packageContext: this@JsonActivity, UserActivity::class.java).also { it: Intent
                startActivity(it)
            }
        }

        btnPost.setOnClickListener { it: View!
            Intent( packageContext: this@JsonActivity, PostActivity::class.java).also { it: Intent
                startActivity(it)
            }
        }
    }
}
```

Langkah 3.5: Buat layout ***Activity*_json.xml** di folder **res/layout/:**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
 xmlns:tools="http://schemas.android.com/tools"
 android:id="@+id/main"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 android:orientation="horizontal"
 android:gravity="center"
 tools:context=".topik9. Activity.Json Activity">

<Button
android:id="@+id/btnUser"
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="User"
android:textSize="16sp" />
<Button
android:id="@+id/btnPost"
```

```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_weight="1"
android:text="POST"
android:textSize="16sp" />
 </LinearLayout>
```

Langkah 3.6: Buat file **Post *Activity*.kt** di dalam package ***Activity*** dengan cara klik kanan pada package, **pilih New > *Activity* > Empty Views *Activity*.**

Langkah 3.7: Pada jendela pengaturan *Activity*, lakukan konfigurasi berikut:

a. Pada opsi ***Activity* Name**, ketik **Post *Activity*.**
b. Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
c. Pada bagian **Layout Name**, ketik ***Activity*_post.xml** sebagai nama layout.
d. Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.
e. Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.topik9**
f. Pada bagian **Source Language**, pastikan bahasa yang dipilih adalah **Kotlin.**

Langkah 3.8: Setelah semua pengaturan sesuai, klik **Finish** untuk membuat *Activity* baru beserta layout-nya.

Langkah 3.9: Tambahkan kode berikut pada **Post *Activity*.kt:**

DUMMY

PostActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik9.activity

import ...

class PostActivity : AppCompatActivity() {

    private lateinit var rvPost: RecyclerView
    private lateinit var tvResponseCode: TextView
    private var list = ArrayList<PostResponse>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_post)

        rvPost = findViewById(R.id.rvPost)
        tvResponseCode = findViewById(R.id.tvResponseCode)
        showPosts()
    }
    private fun showPosts() {
        rvPost.setHasFixedSize(true)
        rvPost.layoutManager = LinearLayoutManager( context: this)
        RetrofitClient.instance.getPosts().enqueue(object : Callback<ArrayList<PostResponse>> {
            override fun onFailure(call: Call<ArrayList<PostResponse>>, t: Throwable) {
                tvResponseCode.text = "Error: ${t.message}"
                Toast.makeText( context: this@PostActivity,  text: "Failed to load posts", Toast.LENGTH_SHORT).show()
            }
            override fun onResponse(
                call: Call<ArrayList<PostResponse>>,
                response: Response<ArrayList<PostResponse>>
            ) {
                tvResponseCode.text = "Response code: ${response.code()}"
                val listResponse = response.body()
                if (listResponse != null && listResponse.isNotEmpty()) {
                    list.clear()
                    list.addAll(listResponse)
                    val adapter = PostAdapter(list)
                    rvPost.adapter = adapter
                } else {
                    Toast.makeText( context: this@PostActivity,  text: "No posts available", Toast.LENGTH_SHORT).show()
                }
            }
        })
    }
}
```

Langkah 3.10: Buat layout ***Activity*_post.xml** di folder **res/layout/:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
 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=".topik10. Activity.Post Activity">

 <TextView
 android:id="@+id/tvResponseCode"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginStart="16dp"
 android:layout_marginTop="16dp"
 android:layout_marginEnd="16dp"
 android:text="Response Code"
```

```
 android:textSize="16sp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

 <androidx.RecyclerView.widget.RecyclerView
 android:id="@+id/rvPost"
 android:layout_width="0dp"
 android:layout_height="0dp"
 android:layout_margin="8dp"

app:layout_constraintTop_toBottomOf="@+id/tvResponseCode
"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintBottom_toBottomOf="parent" />

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

Langkah 3.11: Buat file **User *Activity*.kt** di dalam package ***Activity*** dengan cara klik kanan pada package, pilih **New > *Activity* > Empty Views *Activity*.**

Langkah 3.12: Pada jendela pengaturan *Activity*, lakukan konfigurasi berikut:

a. Pada opsi ***Activity* Name**, ketik **User *Activity***.
b. Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
c. Pada bagian **Layout Name**, ketik ***Activity*_user.xml** sebagai nama layout.
d. Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.
e. Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.topik9**
f. Pada bagian **Source Language**, pastikan bahasa yang dipilih adalah **Kotlin**.

Langkah 3.13: Tambahkan kode berikut pada **User *Activity*.kt:**

DUMMY

UserActivity.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik9.activity

import android.os.Bundle
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.kurniawan.belajarandroidkotlin.R
import com.kurniawan.belajarandroidkotlin.topik9.adapter.UserAdapter
import com.kurniawan.belajarandroidkotlin.topik9.api.RetrofitClient
import com.kurniawan.belajarandroidkotlin.topik9.model.User
import com.kurniawan.belajarandroidkotlin.topik9.model.UserResponse
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class UserActivity : AppCompatActivity() {
    private var list = ArrayList<User>()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_user)
        showUsers()
    }
    private fun showUsers() {
        val rvUser = findViewById<RecyclerView>(R.id.rvUser)
        val tvResponseCode = findViewById<TextView>(R.id.tvResponseCode)

        rvUser.setHasFixedSize(true)
        rvUser.layoutManager = LinearLayoutManager( context: this)

        RetrofitClient.instance.getUsers().enqueue(object : Callback<UserResponse> {
            override fun onFailure(call: Call<UserResponse>, t: Throwable) {
                tvResponseCode.text = t.message
            }
            override fun onResponse(call: Call<UserResponse>, response: Response<UserResponse>) {
                tvResponseCode.text = response.code().toString()
                val listResponse = response.body()?.data
                listResponse?.let { list.addAll(it) }
                val adapter = UserAdapter(list)
                rvUser.adapter = adapter
            }
        })
    }
}
```

Langkah 3.14: Buat layout ***Activity*_user.xml** di folder **res/layout/:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/androi
d"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="match_parent"
 android:layout_height="match_parent"
 tools:context=".topik9. Activity.User Activity">

 <TextView
 android:id="@+id/tvResponseCode"
 android:layout_width="0dp"
```

```
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="16dp"
    android:text="response code"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

    <androidx.RecyclerView.widget.RecyclerView
    android:id="@+id/rvUser"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_margin="8dp"

app:layout_constraintTop_toBottomOf="@id/tvResponseCode"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintBottom_toBottomOf="parent" />

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

4. Implementasi api pada package topik9

Langkah 4.1: Buat data class **Api.kt** di dalam package *topik9* dengan cara cara klik kanan pada package, pilih **New** > **Kotlin Class/File** > **Interface.**

Langkah 4.2: Tambahkan kode berikut pada **Api.kt**:

Api.kt

```
1  package com.kurniawan.belajarandroidkotlin.topik9.api
2
3  import com.kurniawan.belajarandroidkotlin.topik9.model.PostResponse
4  import com.kurniawan.belajarandroidkotlin.topik9.model.UserResponse
5  import retrofit2.Call
6  import retrofit2.http.GET
7
8  interface Api {
9      @GET("users")
10     fun getUsers(): Call<UserResponse>
11
12     @GET("https://jsonplaceholder.typicode.com/posts")
13     fun getPosts(): Call<ArrayList<PostResponse>>
14 }
15
```

Langkah 4.3: Buat data class **RetrofitInstance.kt** di dalam package *topik9* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File > Object**

Langkah 4.4: Tambahkan kode berikut pada **RetrofitInstance.kt:**

RetrofitClient.kt

```
1  package com.kurniawan.belajarandroidkotlin.topik9.api
2
3  import  retrofit2.Retrofit
4  import  retrofit2.converter.gson.GsonConverterFactory
5
6  object RetrofitClient {
7      private  const val BASE_URL = "https://reqres.in/api/"
8
9      val instance: Api by lazy {
10         val retrofit = Retrofit.Builder()  Retrofit.Builder
11             .baseUrl(BASE_URL)  Retrofit.Builder
12             .addConverterFactory(GsonConverterFactory.create())
13             .build()
14
15         retrofit.create(Api::class.java)  ^lazy
16     }
17 }
```

5. Implementasi model pada package topik9

Langkah 5.1: Buat data class **PostResponse.kt** di dalam package *topik9* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File > File.**

Langkah 5.2: Tambahkan kode berikut pada **PostResponse.kt:**

PostResponse.kt

```
1  package com.kurniawan.belajarandroidkotlin.topik9.model
2
3  data class PostResponse(
4      val userId: Int,
5      val id: Int,
6      val title: String
7  )
```

Langkah 5.3: Buat data class **User.kt** di dalam package *topik9* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File > File.**

Langkah 5.4: Tambahkan kode berikut pada **User.kt:**

User.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik9.model

import com.google.gson.annotations.SerializedName

data class User(
    val id: Int,
    val email: String,
    @SerializedName("first_name")
    val firstName: String,
    @SerializedName("last_name")
    val lastName: String
)
```

Langkah 5.5: Buat data class **UserResponse.kt** di dalam package *topik9* dengan cara cara klik kanan pada package, pilih **New > Kotlin Class/File > File.**

Langkah 5.6: Tambahkan kode berikut pada **UserResponsekt:**

UserResponse.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.topik9.model

data class UserResponse(
  val data: ArrayList<User>
 )
```

6. Membuat list item

Langkah 6.1: Buat layout baru dengan name **item**_list di dalam package layout dengan cara cara klik kanan pada layout, pilih **New > Layout *resource* File**

Langkah 6.2: Tambahkan kode berikut pada **item_list.xml:**

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

 <TextView
 android:id="@+id/tvResponse"
```

```
android:layout_width="0dp"
android:layout_height="wrap_content"
android:layout_margin="16dp"
android:text="Response"
android:textSize="16sp"
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"/>

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

7. Mengkonfigurasi AndroidManifest.xml

Langkah 7.1: Tambahkan deklarasi setaip *Activity* di **AndroidManifest.xml** untuk mengenali file-file ini dalam aplikasi:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
    <uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BelajarAndroidKotlin"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        tools:targetApi="31">

        <activity
            android:name=".topik9.activity.PostActivity"
            android:exported="true" />

        <activity
            android:name=".topik9.activity.UserActivity"
            android:exported="false" />
```

8. Menjalankan Dan Menguji Aplikasi

Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.

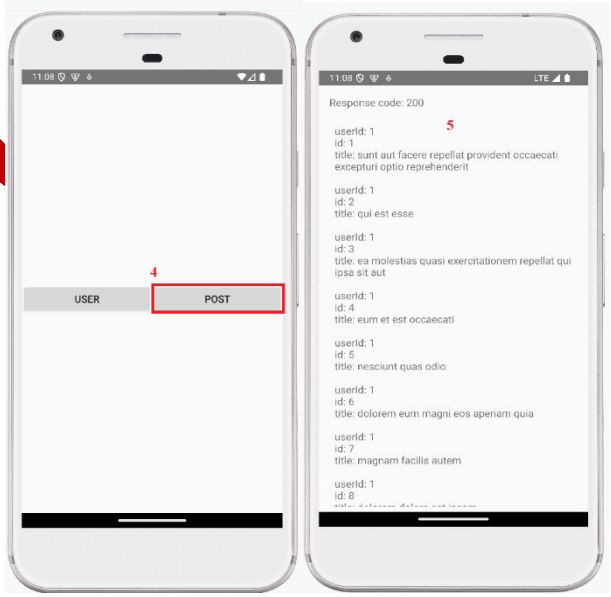
Langkah 8.1: Klik **Run** pada toolbar Android Studio atau tekan **Shift + F10** untuk memulai proses kompilasi dan menjalankan aplikasi.

Langkah 8.2: Pada jendela **Select Deployment Target**, pilih perangkat yang akan digunakan untuk menguji aplikasi:

a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode **USB Debugging** aktif.

9. Hasil Pengujian

Setelah aplikasi berhasil dijalankan, Anda akan melihat tampilan berikut pada layar perangkat atau emulator:



Gambar 16. Hasil Pengujian Web Service Json Object & Json Array HTTP Request dengan Retrofit

E. Tugas

1. **Modifikasi** : Modifikasi aplikasi yang telah anda kerjakan dan buatlah halaman sederhana yang menampilkan daftar produk dari sebuah web service. Silahkan temukan api baru yang menampilkan nama produk.
2. **Laporan** : Buat screenshot hasil implementasi dan berikan penjelasan singkat untuk setiap langkah yang sudah dilakukan.

DUMMY

BAB 12
PROJECT 1

A. Membangun Aplikasi Menu Makanan dengan Kotlin

Aplikasi Menu Makanan merupakan sebuah aplikasi berbasis Android yang dikembangkan menggunakan Android Studio dengan bahasa pemrograman Kotlin. Aplikasi ini dirancang untuk memberikan kemudahan bagi pengguna dalam menelusuri berbagai menu makanan dan minuman yang tersedia dengan tampilan yang modern dan interaktif. Dalam aplikasi ini, terdapat beberapa komponen utama yang mendukung kemudahan navigasi dan pencarian, yaitu:

1. ***RecyclerView*** – digunakan untuk menampilkan daftar menu secara dinamis dalam bentuk kartu yang dapat di-scroll, memudahkan pengguna untuk melihat beragam pilihan makanan dan minuman.
2. **SearchView** – memungkinkan pengguna untuk mencari menu secara langsung dengan memasukkan kata kunci, sehingga proses pencarian menu menjadi lebih cepat dan efisien.
3. **Intent** – memfasilitasi navigasi antar halaman, seperti saat pengguna ingin melihat detail suatu menu tertentu atau mengakses informasi lainnya di aplikasi.
4. **TabHost** – digunakan untuk membagi tampilan menu menjadi beberapa kategori, seperti Makanan dan Minuman, sehingga pengguna dapat dengan mudah beralih antar kategori menu yang diinginkan.

Aplikasi ini dirancang untuk memberikan pengalaman pengguna yang baik dan praktis dalam mencari dan memilih makanan atau minuman, serta memanfaatkan berbagai fitur Android untuk meningkatkan fungsionalitas dan kenyamanan.

B. Langkah-Langkah Pembuatan Aplikasi

1. **Persiapan Proyek Android Studio**
   Langkah 1.1: Buka Android Studio
   Langkah 1.2: Beri nama package project1 dengan cara klik kanan pada package **com.kurniawan.belajarandroidkotlin** lalu klik **New > package >**

DUMMY

2. **Memperbarui code pada Main *Activity*.kt**

   Langkah 2.1: Buka file **Main *Activity*.kt** yang secara otomatis sudah ada di dalam folder **kotlin+java/com.kurniawan.belajarandroidkotlin**.

   Langkah 2.2: Tambahkan kode berikut untuk mendefinisikan tombol navigasi *Activity* Home *Activity*

```
val button10 = findViewById<Button>(R.id.button10)
button10.setOnClickListener {
    val intent = Intent( packageContext: this, HomeActivity::class.java)
    startActivity(intent)
}
```

3. **Mengatur layout *Activity*_main.xml**

   Langkah 3.1: Buka file ***Activity*_main.xml** di dalam folder **res/layout/.**

   Langkah 3.2: Tambahkan kode XML berikut untuk membuat tampilan button Weather Page pada layout.

```
<Button
    android:id="@+id/button10"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Projek 1 - Menu Food" />
```

4. **Membuat *Activity* Baru**

   Langkah 4.1: Membuat *Activity* baru sebagai halaman utama dengan nama Home *Activity* dan buat seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

import android.content.Intent
import androidx.appcompat.app.AppCompat Activity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import android.widget.ImageView
import com.kurniawan.belajarandroidkotlin.R

class Home Activity : AppCompat Activity() {
 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout. Activity_home)

 val welcomeText: TextView =
```

```
findViewById(R.id.welcomeText)
 welcomeText.text = "Selamat datang di Menu Food! Temukan
berbagai pilihan makanan favorit Anda di sini."

 val foodLogo: ImageView = findViewById(R.id.foodLogo)

 val foodButton: Button = findViewById(R.id.foodButton)
 foodButton.text = "Lihat Menu Makanan"
 foodButton.setOnClickListener {
  val intent = Intent(this, Food Activity::class.java)
  start Activity(intent)
 }
 }
}
```

Langkah 4.2: Membuat halaman tampilan list menu dengan nama Food *Activity* dan buat seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

import android.os.Bundle
import androidx.appcompat.widget.SearchView
import android.widget.TabHost
import androidx.appcompat.app.AppCompat Activity
import androidx.RecyclerView.widget.LinearLayoutManager
import androidx.RecyclerView.widget.RecyclerView
import com.kurniawan.belajarandroidkotlin.R

class Food Activity : AppCompat Activity() {
 private lateinit var makananAdapter: MenuAdapter
 private lateinit var minumanAdapter: MenuAdapter

 override fun onCreate(savedInstanceState: Bundle?) {
 super.onCreate(savedInstanceState)
 setContentView(R.layout. Activity_food)

 val tabHost = findViewById<TabHost>(R.id.tabHost)
 tabHost.setup()

 val                    searchView                    =
findViewById<SearchView>(R.id.searchView)

 // Menambahkan Tab Makanan
 val tabSpecMakanan = tabHost.newTabSpec("Makanan")
 tabSpecMakanan.setIndicator("Makanan")
 tabSpecMakanan.setContent(R.id.tabMakanan)
 tabHost.addTab(tabSpecMakanan)
```

```
// Menambahkan Tab Minuman
val tabSpecMinuman = tabHost.newTabSpec("Minuman")
tabSpecMinuman.setIndicator("Minuman")
tabSpecMinuman.setContent(R.id.tabMinuman)
tabHost.addTab(tabSpecMinuman)

// Daftar Menu Makanan
val makananList = listOf(
 MenuItem("Nasi Goreng", 15000.0),
 MenuItem("Ayam Bakar", 20000.0),
 MenuItem("Sate Padang", 18000.0),
 MenuItem("Mie Goreng", 12000.0),
 MenuItem("Pecel Lele", 13000.0),
 MenuItem("Nasi Uduk", 16000.0),
 MenuItem("Soto Ayam", 25000.0),
 MenuItem("Bakso", 22000.0),
 MenuItem("Rendang", 30000.0),
 MenuItem("Gado-Gado", 14000.0),
 MenuItem("Nasi Liwet", 22000.0),
 MenuItem("Ayam Penyet", 18000.0),
 MenuItem("Ikan Bakar", 27000.0),
 MenuItem("Tahu Tempe", 8000.0),
 MenuItem("Karedok", 15000.0),
 MenuItem("Tongseng", 23000.0),
 MenuItem("Martabak Telur", 25000.0),
 MenuItem("Bakmi Noodles", 17000.0),
 MenuItem("Pisang Goreng", 9000.0)
)

// Daftar Menu Minuman
val minumanList = listOf(
 MenuItem("Es Teh", 5000.0),
 MenuItem("Jus Jeruk", 10000.0),
 MenuItem("Kopi Susu", 7000.0),
 MenuItem("Es Kelapa Muda", 12000.0),
 MenuItem("Jus Apel", 9000.0),
 MenuItem("Teh Tarik", 8000.0),
 MenuItem("Lemon Tea", 7500.0),
 MenuItem("Milkshake", 15000.0),
 MenuItem("Air Mineral", 3000.0),
 MenuItem("Cokelat Panas", 11000.0),
 MenuItem("Es Krim", 15000.0),
 MenuItem("Soda Gembira", 10000.0),
 MenuItem("Es Jeruk Nipis", 8500.0),
 MenuItem("Jus Tomat", 9500.0),
 MenuItem("Es Campur", 12000.0),
 MenuItem("Smoothie Bowl", 18000.0),
```

```
    MenuItem("Bubble Tea", 13000.0),
    MenuItem("Susu Cokelat", 10000.0),
    MenuItem("Jus Mangga", 11000.0),
    MenuItem("Minuman Jahe", 7000.0)
  )

  // Mengatur RecyclerView untuk Tab Makanan
  val makananRecyclerView =
findViewById<RecyclerView>(R.id.recyclerMakanan)
  makananRecyclerView.layoutManager =
LinearLayoutManager(this)
  makananAdapter = MenuAdapter(this, makananList)
  makananRecyclerView.Adapter = makananAdapter

  // Mengatur RecyclerView untuk Tab Minuman
  val minumanRecyclerView =
findViewById<RecyclerView>(R.id.recyclerMinuman)
  minumanRecyclerView.layoutManager =
LinearLayoutManager(this)
  minumanAdapter = MenuAdapter(this, minumanList)
  minumanRecyclerView.Adapter = minumanAdapter

  // Fungsi pencarian
  searchView.setOnQueryTextListener(object :
SearchView.OnQueryTextListener {
    override fun onQueryTextSubmit(query: String?): Boolean
{
    return false
    }

    override fun onQueryTextChange(newText: String?):
Boolean {
    val filteredMakanan = makananList.filter {
it.name.contains(newText ?: "", ignoreCase = true) }
    val filteredMinuman = minumanList.filter {
it.name.contains(newText ?: "", ignoreCase = true) }

    makananAdapter.updateData(filteredMakanan)
    minumanAdapter.updateData(filteredMinuman)

    return true
    }
  })
  }
}
```

Langkah 4.3: Membuat *Adapter* nya dengan nama Menu*Adapter* dan buat seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.RecyclerView.widget.RecyclerView
import com.kurniawan.belajarandroidkotlin.R

class MenuAdapter(
 private val context: Context,
 private var itemList: List<MenuItem>
) : RecyclerView.Adapter<MenuAdapter.ViewHolder>() {

 class ViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
 val nameText: TextView =
itemView.findViewById(R.id.menu_name)
 val priceText: TextView =
itemView.findViewById(R.id.menu_price)
 }

 override fun onCreateViewHolder(parent: ViewGroup,
viewType: Int): ViewHolder {
 val view =
LayoutInflater.from(context).inflate(R.layout.
Activity_menu_item, parent, false)
 return ViewHolder(view)
 }

 override fun onBindViewHolder(holder: ViewHolder,
position: Int) {
 val menuItem = itemList[position]
 holder.nameText.text = menuItem.name
 holder.priceText.text = "Rp ${menuItem.price}"
 }

 override fun getItemCount(): Int = itemList.size

 fun updateData(newItemList: List<MenuItem>) {
 itemList = newItemList
 notifyDataSetChanged()
 }
}
```

Langkah 4.4: Membuat *Activity* baru dengan nama MenuItem untuk mengisi item pada *Adapter* dan buat seperti berikut:

```
package com.kurniawan.belajarandroidkotlin

data class MenuItem(val name: String, val price: Double)
```

5. **Menambahkan komponen UI**

Langkah 5.1: Isikan kode xml ke ***Activity_home*** dengan Root Tag LinearLayout seperti berikut:

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

 <ImageView
 android:id="@+id/foodLogo"
 android:layout_width="376dp"
 android:layout_height="226dp"
 android:layout_marginTop="68dp"
 android:src="@drawable/food_logo"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.485"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toTopOf="parent" />

 <TextView
 android:id="@+id/welcomeText"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_marginTop="8dp"
 android:gravity="center"
 android:padding="25dp"
 android:text="Selamat datang di Menu Food! Temukan
berbagai pilihan makanan favorit Anda di sini."
 android:textSize="18sp"
 app:layout_constraintEnd_toEndOf="parent"
 app:layout_constraintHorizontal_bias="0.0"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintTop_toBottomOf="@id/foodLogo" />
```

```
 <Button
 android:id="@+id/foodButton"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Lihat Menu Makanan"
 android:layout_marginTop="32dp"
 app:layout_constraintTop_toBottomOf="@id/welcomeText"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Langkah 5.2: Isikan kode xml ke ***Activity_food*** dengan Root Tag LinearLayout seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout

xmlns:android="http://schemas.android.com/apk/res/android
"
 xmlns:app="http://schemas.android.com/apk/res-auto"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <androidx.appcompat.widget.SearchView
 android:id="@+id/searchView"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_marginTop="16dp"
 android:queryHint="Cari menu makanan atau minuman"
 app:layout_constraintTop_toTopOf="parent"
 app:layout_constraintStart_toStartOf="parent"
 app:layout_constraintEnd_toEndOf="parent"/>

 <TabHost
 android:id="@+id/tabHost"
 android:layout_width="match_parent"
 android:layout_height="0dp"
 app:layout_constraintTop_toBottomOf="@id/searchView"
 app:layout_constraintBottom_toBottomOf="parent">

 <LinearLayout
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">

  <TabWidget
  android:id="@android:id/tabs"
  android:layout_width="match_parent"
```

```
  android:layout_height="wrap_content"/>

  <FrameLayout
  android:id="@android:id/tabcontent"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <!-- Layout for Tab Makanan -->
  <LinearLayout
   android:id="@+id/tabMakanan"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <androidx.RecyclerView.widget.RecyclerView
   android:id="@+id/recyclerMakanan"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
  </LinearLayout>

  <!-- Layout for Tab Minuman -->
  <LinearLayout
   android:id="@+id/tabMinuman"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical">
   <androidx.RecyclerView.widget.RecyclerView
   android:id="@+id/recyclerMinuman"
   android:layout_width="match_parent"
   android:layout_height="match_parent"/>
  </LinearLayout>

   </FrameLayout>
  </LinearLayout>
  </TabHost>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Langkah 5.3: Isikan kode xml ke ***Activity_menu_Adapter*** dengan Root Tag LinearLayout seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/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=".MenuAdapter">
```

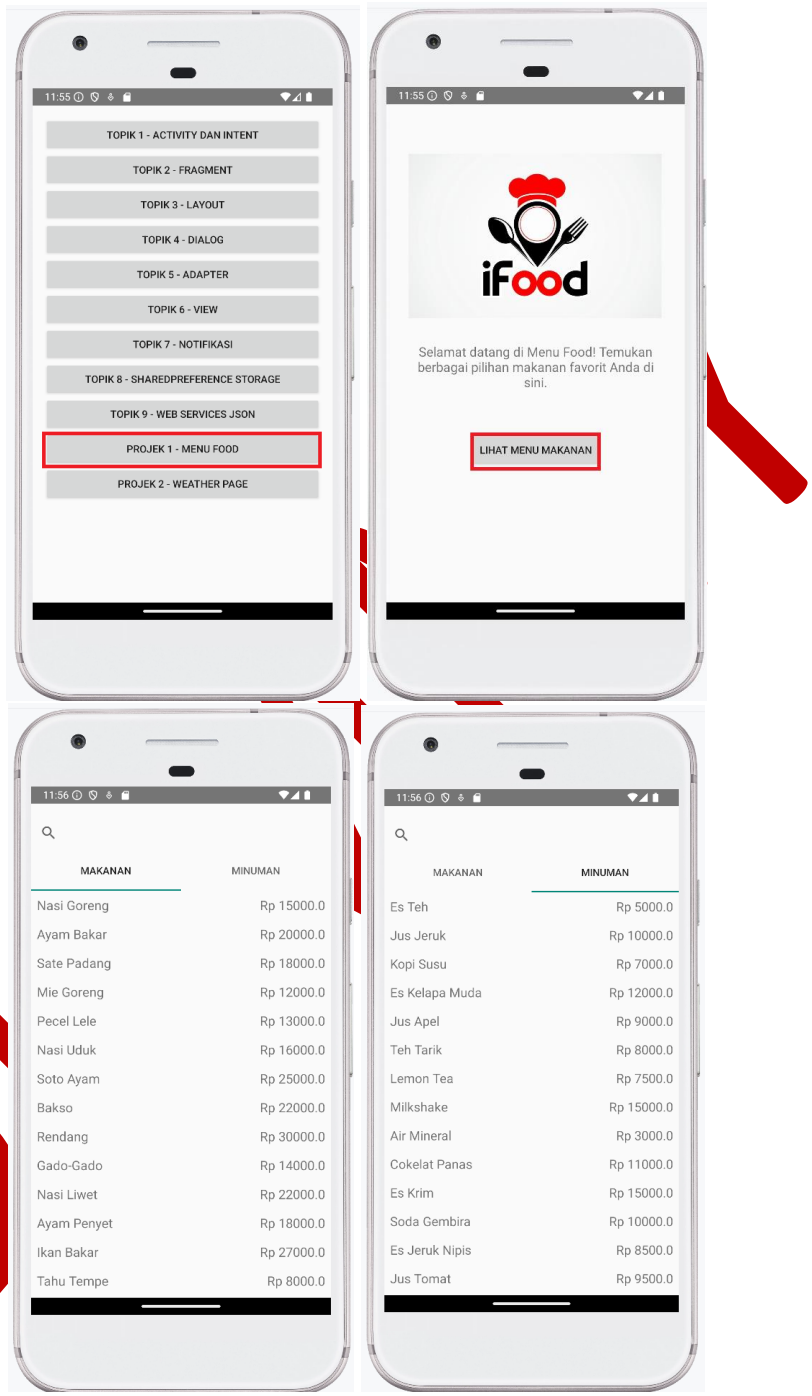
```
</androidx.constraintlayout.widget.ConstraintLayout>
```

Langkah 5.4: Isikan kode xml ke ***Activity_menu_item*** dengan Root Tag LinearLayout seperti berikut:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:orientation="horizontal"
 android:padding="8dp">

 <TextView
 android:id="@+id/menu_name"
 android:layout_width="0dp"
 android:layout_height="wrap_content"
 android:layout_weight="1"
 android:textSize="18sp"/>

 <TextView
 android:id="@+id/menu_price"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:textSize="18sp"/>
</LinearLayout>
```

6. **Menjalankan dan Menguji Aplikasi**
   Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.
   Langkah 1: Klik Run pada toolbar Android Studio atau tekan **Shift + F10** untuk memulai proses kompilasi dan menjalankan aplikasi.
   Langkah 2: Pada jendela **Select Deployment Target**, pilih perangkat yang akan digunakan untuk menguji aplikasi:
   Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
   Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode **USB Debugging** aktif.
7. **Hasil Pengujian**

DUMMY



Gambar 17. Hasil Pengujian Project 1

BAB 13
PROJECT 2

A. Membangun Aplikasi Cuaca Dengan Retrofit

Aplikasi yang dibangun merupakan sebuah aplikasi cuaca berbasis Android yang memanfaatkan Retrofit untuk menarik data cuaca dari API eksternal. Aplikasi ini memungkinkan pengguna mencari informasi cuaca terkini berdasarkan lokasi yang diinginkan, dan menampilkan data seperti suhu, kelembapan, kecepatan angin, serta kondisi cuaca dengan antarmuka yang ramah pengguna.

Proses pengembangan dimulai dari konfigurasi proyek di Android Studio, dengan struktur paket yang rapi untuk komponen-komponen seperti tampilan utama, aktivitas cuaca, dan layout XML yang disesuaikan. Implementasi logika aplikasi dilakukan melalui Main *Activity*, di mana pengguna dapat bernavigasi ke halaman cuaca dengan menekan tombol yang telah disediakan.

Untuk integrasi API, modul api mengelola komunikasi dengan server cuaca menggunakan Retrofit, sedangkan JSON response dari server diubah menjadi kelas data Kotlin yang kemudian disajikan dalam antarmuka aplikasi. Pada WeatherPage, informasi cuaca diperlihatkan dengan dukungan elemen UI seperti ikon lokasi, indikator suhu, dan rincian waktu. Setelah implementasi dan konfigurasi di *Gradle* serta AndroidManifest.xml, aplikasi diuji pada perangkat fisik atau emulator, memastikan bahwa data cuaca tampil dengan akurat sesuai masukan pengguna.Ketik konten Sub Bab disini

DUMMY

B. Langkah-Langkah Implementasi Aplikasi

1. Persiapan Proyek Android Studio
   Langkah 1.1: Buka Android Studio
   Langkah 1.2: Beri nama package **project2** dengan cara klik kanan pada package *com.kurniawan.belajarandroidkotlin* lalu klik **New > package >**
2. Memperbarui code pada **Main *Activity*.kt**
   Langkah 2.1: Buka file **Main *Activity*.kt** yang secara otomatis sudah ada di dalam folder **kotlin+java/com.kurniawan.belajarandroidkotlin.**
   Langkah 2.2: Tambahkan kode berikut untuk mendefinisikan tombol navigasi *Activity* Weather Page.

```
val buttonWeatherPage = findViewById<Button>(R.id.buttonWeatherPage)
buttonWeatherPage.setOnClickListener { it: View!
    val intent = Intent( packageContext: this, WeatherActivity::class.java)
    startActivity(intent)
}
```

3. Mengatur layout *Activity*_main.xml
   Langkah 3.1: Buka file ***Activity*_main.xml** di dalam folder **res/layout/.**
   Langkah 3.2: Tambahkan kode XML berikut untuk membuat tampilan button Weather Page pada layout

```
<Button
    android:id="@+id/buttonWeatherPage"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Weather Page" />
```

4. Implementasi aplikasi cuaca dengan Menggunakan Retrofit
   Langkah 4.1: Buat file **Weather *Activity*.kt** di dalam package **project2** dengan cara klik kanan pada package, pilih **New > *Activity* > Empty Views *Activity*.**
   Langkah 4.2: Pada jendela pengaturan *Activity*, lakukan konfigurasi berikut:
   a. Pada opsi ***Activity* Name**, ketik **Weather *Activity*.**
   b. Centang opsi **Generate a Layout File** untuk membuat file layout otomatis.
   c. Pada bagian **Layout Name**, ketik ***Activity*_weather.xml** sebagai nama layout.
   d. Abaikan bagian **Launcher *Activity*** karena tidak perlu diubah.
   e. Pastikan **Package Name** berada pada package **com.kurniawan.belajarandroidkotlin.project2**
   f. Pada bagian Source Language, pastikan bahasa yang dipilih adalah Kotlin.
   Langkah 4.3: Setelah semua pengaturan sesuai, klik **Finish** untuk membuat *Activity* baru beserta layout-nya.
   Langkah 4.4: Tambahkan kode berikut pada **Weather *Activity*.kt:**

DUMMY

WeatherActivity.kt

```
package com.kurniawan.belajarandroidkotlin.project2

import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.lifecycle.ViewModelProvider

class WeatherActivity : ComponentActivity() {

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

        val weatherViewModel = ViewModelProvider( owner: this)[WeatherViewModel::class.java]

        setContent {
            WeatherPage(weatherViewModel)
        }
    }
}

```

Langkah 4.5: Buat layout ***Activity*_weather.xml** di folder **res/layout/** untuk **Weather *Activity*:**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/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=".project2.Weather Activity">

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

5. Implementasi Weather Page

Langkah 5.1: Buat file **WeatherPage.kt** di dalam package **project2** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File.**

Langkah 5.2: Tambahkan kode berikut pada **WeatherPage.kt:**

```
package com.kurniawan.belajarandroidkotlin.project2

import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Row
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.layout.width
```

```
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.LocationOn
import androidx.compose.material.icons.filled.Lock
import androidx.compose.material.icons.filled.Search
import androidx.compose.material3.Card
import androidx.compose.material3.CardDefaults
import
androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.Icon
import androidx.compose.material3.IconButton
import androidx.compose.material3.OutlinedTextField
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.livedata.observeAsState
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.runtime.setValue
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import coil3.compose.AsyncImage
import
com.kurniawan.belajarandroidkotlin.api.NetworkResponse
import
com.kurniawan.belajarandroidkotlin.api.WeatherModel

@Composable
fun WeatherPage(viewModel: WeatherViewModel) {

 var city by remember {
 mutableStateOf("")
 }
 val weatherResult =
viewModel.weatherResult.observeAsState()
 Column(
 modifier = Modifier
  .fillMaxWidth()
  .padding(16.dp),
 horizontalAlignment = Alignment.CenterHorizontally
 ) {
 Row(
  modifier = Modifier
```

```
    .fillMaxWidth()
    .padding(8.dp),
    verticalAlignment = Alignment.CenterVertically,
    horizontalArrangement = Arrangement.SpaceEvenly
   ) {
    OutlinedTextField(
    modifier = Modifier.weight(1f),
    value = city,
    onValueChange = {
     city = it
    },
    label = {
     Text(text = "Search For Any Location")
    }
    )
    IconButton(onClick = {
    viewModel.getData(city) }) {
    Icon(
     imageVector = Icons.Default.Search,
     contentDescription = "Search For Any Location"
    )
    }
   }
   when(val result = weatherResult.value){
    is NetworkResponse.Eror -> {
    Text(text = result.message, color = Color.Red)
    }
    is NetworkResponse.Loading -> {
    CircularProgressIndicator()
    }
    is NetworkResponse.Success -> {
    WeatherDetails(data = result.data)
    }
    null -> {}
   }
   }
  }
  @Composable
  fun WeatherDetails(data: WeatherModel) {
   Column(
   modifier = Modifier
    .fillMaxWidth()
    .padding(vertical = 8.dp),
   horizontalAlignment = Alignment.CenterHorizontally
   ) {
   Row(
    modifier = Modifier.fillMaxWidth(),
```

```
    horizontalArrangement = Arrangement.Start,
    verticalAlignment = Alignment.Bottom
   ) {
    Icon(
    imageVector = Icons.Default.LocationOn,
    contentDescription = "Location icon",
    modifier = Modifier.size(40.dp)
    )
    Spacer(modifier = Modifier.width(8.dp))
    Column {
    Text(text = data.location.name, fontSize = 30.sp,
fontWeight = FontWeight.Bold)
    Text(
     text = data.location.country,
     fontSize = 18.sp,
     color = Color.Gray
    )
    }
   }

   Spacer(modifier = Modifier.height(16.dp))
   Text(
    text = "${data.current.temp_c} °C",
    fontSize = 56.sp,
    fontWeight = FontWeight.Bold,
    textAlign = TextAlign.Center
   )
   Text(
    text = data.current.condition.text,
    fontSize = 20.sp,
    textAlign = TextAlign.Center,
    color = Color.Gray
   )
   Spacer(modifier = Modifier.height(16.dp))
   Card(
    modifier = Modifier
    .fillMaxWidth()
    .padding(16.dp),
    elevation = CardDefaults.cardElevation(defaultElevation
= 4.dp)
   ) {
    Column(
    modifier = Modifier.fillMaxWidth()
    ) {
    Row(
     modifier = Modifier.fillMaxWidth(),
     horizontalArrangement = Arrangement.SpaceAround
```

```
    ) {
     WeatherKeyVal("Humidity", "${data.current.humidity}
%")
     WeatherKeyVal("Wind Speed", "${data.current.wind_kph}
km/h")
    }
    Row(
     modifier = Modifier.fillMaxWidth(),
     horizontalArrangement = Arrangement.SpaceAround
    ) {
     WeatherKeyVal("UV", data.current.uv.toString())
     WeatherKeyVal("Precipitation",
"${data.current.precip_mm} mm")
    }
    Row(
     modifier = Modifier.fillMaxWidth(),
     horizontalArrangement = Arrangement.SpaceAround
    ) {
     WeatherKeyVal("Local Time",
data.location.localtime.split(" ")[1])
     WeatherKeyVal("Local Date",
data.location.localtime.split(" ")[0])
    }
    }
   }
   }
}
@Composable
fun WeatherKeyVal(key: String, value: String) {
 Column(
 modifier = Modifier.padding(16.dp),
 horizontalAlignment = Alignment.CenterHorizontally
 ) {
 Text(text = value, fontSize = 24.sp, fontWeight =
FontWeight.Bold)
 Text(text = key, fontWeight = FontWeight.SemiBold, color
= Color.Gray)
 }
}
```

6. Implementasi View Model

Langkah 6.1: Buat file **WeatherViewModel.kt** di dalam package **project2** dengan cara klik kanan pada package, pilih **New** > **Kotlin Class/File.**

Langkah 6.2: Tambahkan kode berikut pada **WeatherViewModel.kt:**

WeatherViewModel.kt

```
package com.kurniawan.belajarandroidkotlin.project2

import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.kurniawan.belajarandroidkotlin.api.Constant
import com.kurniawan.belajarandroidkotlin.api.NetworkResponse
import com.kurniawan.belajarandroidkotlin.api.RetrofitInstance
import com.kurniawan.belajarandroidkotlin.api.WeatherModel
import kotlinx.coroutines.launch
import java.lang.Exception

class WeatherViewModel :ViewModel() {

    private val weatherApi = RetrofitInstance.weatherApi
    private val _weatherResult = MutableLiveData<NetworkResponse<WeatherModel>>()
    val weatherResult: LiveData<NetworkResponse<WeatherModel>> = _weatherResult


    fun getData(city: String) {
        _weatherResult.value = NetworkResponse.Loading
        viewModelScope.launch { this: CoroutineScope
            try {
                val response = weatherApi.getWeather(Constant.apiKey, city)
                if (response.isSuccessful) {
                    response.body()?.let { it: WeatherModel
                        _weatherResult.value = NetworkResponse.Success(it)
                    }
                } else {
                    _weatherResult.value = NetworkResponse.Error("Failed to load data")
                }
            }
            catch (e : Exception){
                _weatherResult.value = NetworkResponse.Error("Failed to load data")
            }
        }
    }
}
```

7. Implementasi API untuk aplikasi cuaca

Langkah 7.1: Buat package **api** di dalam **com.kurniawan.belajarandroidkotlin** dengan cara klik kanan pada package, pilih **New > Package > Empty.**

Langkah 7.2: Navigasikan ke menu di Android Studio, pilih **File > Settings > Plugins.**

Langkah 7.3: Cari plugin **JSON To Kotlin Class** dan instal plugin tersebut untuk mempermudah konversi JSON ke kelas data Kotlin.

Langkah 7.4: Buka situs **WeatherAPI** dan lakukan registrasi akun jika belum memiliki.Setelah registrasi, Anda akan mendapatkan **API Key** yang akan digunakan untuk mengakses data cuaca.
Langkah 7.5: Masuk ke **API Explorer** di situs WeatherAPI.
Langkah 7.6: Atur konfigurasi sebagai berikut:

a. **Your API Key:** Masukkan API Key yang Anda dapatkan pada langkah sebelumnya.
b. **Protocol:** Pilih **HTTPS** untuk keamanan data**.**
c. **Format:** Pilih **JSON** untuk format data yang lebih mudah diolah.

Langkah 7.7: Setelah konfigurasi selesai, klik **Show Response** untuk melihat data cuaca dari API.
Langkah 7.8: Salin **Response Body** yang ditampilkan**.**
Langkah 7.9: Kembali ke Android Studio, di package **api**, klik kanan, lalu pilih **New > Kotlin Data Class File From JSON.**
Langkah 7.10: Tempel **Response Body** yang telah disalin berikan nama class sebagai WeatherModel. Proses ini akan menghasilkan beberapa file model seperti **Current, Location,** dan **Condition** yang akan mewakili data dari JSON response.

8. Implementasi konten package api

Langkah 8.1: Buka file **Condition.kt**
Langkah 8.2: Tambahkan kode berikut pada **Condition.kt:**

Condition.kt

```
1  package com.kurniawan.belajarandroidkotlin.api
2
3  data class Condition(
4      val code: String,
5      val icon: String,
6      val text: String
7  )
```

Langkah 8.5: Buka file **Location.kt**
Langkah 8.6: Tambahkan kode berikut pada **Location.kt:**

CONFIDENTIAL

Location.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.api

data class Location(
    val country: String,
    val lat: String,
    val localtime: String,
    val localtime_epoch: String,
    val lon: String,
    val name: String,
    val region: String,
    val tz_id: String
)
```

Langkah 8.7: Buat file **NetworkResponse.kt** di dalam package **api** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File.**

Langkah 8.8: Tambahkan kode berikut pada **NetworkResponse.kt:**

NetworkResponse.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.api

//T refers to WeatherModel
sealed class NetworkResponse<out T> {
    data class Success<out T>(val data: T ) : NetworkResponse<T>()
    data class Eror(val message : String) : NetworkResponse<Nothing>()
    object Loading : NetworkResponse<Nothing>()
}
```

Langkah 8.9: Buat file **WeatherModel.kt** di dalam package **api** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File.**

Langkah 8.10: Tambahkan kode berikut pada **WeatherModel.kt:**

WeatherModel.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.api

data class WeatherModel(
    val current: Current,
    val location: Location
)
```

Langkah 8.11: Buat file **WeatherApi.kt** di dalam package **api** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File > Interface**

Langkah 8.12: Tambahkan kode berikut pada **WeatherApi.kt:**

WeatherApi.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.api

import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query

interface WeatherApi {
    @GET("/v1/current.json")
    suspend fun getWeather(
        @Query("key") apikey: String,
        @Query("q") city: String
    ): Response<WeatherModel>
}
```

Langkah 8.13: Buat file **Constant.kt** di dalam package **api** dengan cara klik kanan pada package, **pilih New > Kotlin Class/File > Object**
Langkah 8.14: Tambahkan kode berikut pada **Constant.kt:**

Constant.kt

```kotlin
package com.kurniawan.belajarandroidkotlin.api

object Constant {
    val apiKey = "50087a93f0db4d8baf7131108240911"
}
```

Langkah 8.15: Buat file **RetrofitInstance.kt** di dalam package **api** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File > Object**
Langkah 8.16: Tambahkan kode berikut pada **RetrofitInstance.kt:**
Langkah 8.17: Buat file **RetrofitInstance.kt** di dalam package **api** dengan cara klik kanan pada package, pilih **New > Kotlin Class/File > Object**

Langkah 8.18: Tambahkan kode berikut pada **RetrofitInstance.kt:**

RetrofitInstance.kt

```
package com.kurniawan.belajarandroidkotlin.api

import  retrofit2.Retrofit
import  retrofit2.converter.gson.GsonConverterFactory
import retrofit2.create

object RetrofitInstance {

    private const val baseUrl = "https://api.weatherapi.com";

    private fun getInstsance() : Retrofit {
        return Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }

    val weatherApi : WeatherApi = getInstsance().create(WeatherApi::class.java)
}
```

9. Implementasi pada *Gradle*

Langkah 9.1: Buka *Gradle* Script

Langkah 9.2: Tambahkan kode berikut pada bagian paling bawah dari ***Build. Gradle* .kts**

```
val retrofitVersion = "2.11.0"
implementation ("com.squareup.retrofit2:retrofit:$retrofitVersion")
implementation ("com.squareup.retrofit2:converter-gson:$retrofitVersion")
implementation("androidx.compose.runtime:runtime-livedata:1.7.0")
implementation("io.coil-kt.coil3:coil-compose:3.0.1")
```

10. Implementasi pada Manifest

Langkah 10.1: Buka AndroidManifest

Langkah 10.2: Tambahkan deklarasi setiap *Activity* di **AndroidManifest.xml** untuk mengenali file-file ini dalam aplikasi:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="BelajarAndroidKotlin"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.AppCompat.Light.NoActionBar"
        tools:targetApi="31">
        <activity
            android:name=".project2.WeatherActivity"
            android:exported="false" />
```

11. Menjalankan dan Menguji Aplikasi

    Setelah menyelesaikan semua langkah pengkodean dan konfigurasi, saatnya menjalankan aplikasi untuk melihat hasilnya.

    Langkah 11.1: Klik **Run** pada toolbar Android Studio atau tekan **Shift + F10** untuk memulai proses kompilasi dan menjalankan aplikasi.

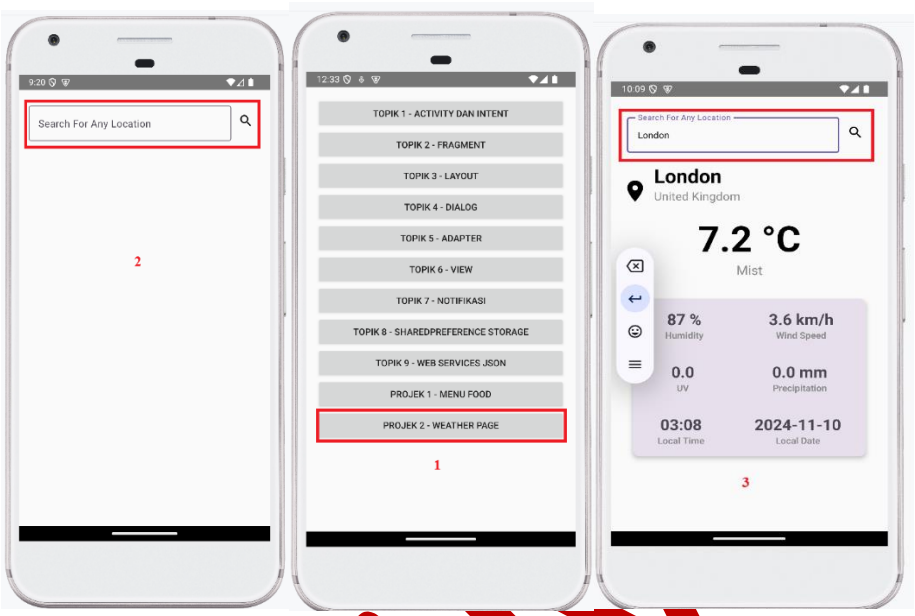
    Langkah 11.2: Pada jendela **Select Deployment** Target, pilih perangkat yang akan digunakan untuk menguji aplikasi:

    a. Jika Anda ingin menggunakan emulator, pastikan emulator Android telah dikonfigurasi dan pilih dari daftar yang tersedia.
    b. Jika menggunakan perangkat fisik, pastikan perangkat telah terhubung ke komputer dan mode USB Debugging aktif.

12. Hasil Pengujian

    Setelah aplikasi berhasil dijalankan, Anda akan melihat tampilan berikut pada layar perangkat atau emulator:

DUMMY



Gambar 18. Hasil Pengujian Project 2

DUMMY

DAFTAR PUSTAKA

Dewi, Ika Parma, Mursyida, Lativa, & Samala, Agariadne Dwinggo. (2021). Dasar-Dasar Android Studio dan Membuat Aplikasi Mobile Sederhana. Widina Bhakti Persada Bandung. ISBN 978-623-6092-37-8.

Google. (2024). CardView. Diakses pada 7 November 2024, dari https://developer.android. com/reference/androidx/cardview/widget/CardView

Google. (2024). Create dynamic lists with *RecyclerView*. Diakses pada 6 November 2024, dari https://developer.android.com/develop/ui/views/layout/*RecyclerView*?hl=id

Google. (2024). *Fragment*. Diakses pada 5 November 2024, dari https://developer.android.com/ guide/components/*Fragment*s?hl=id

Google. (2024). Memahami Siklus Proses Aktivitas. Diakses pada 4 November 2024, dari https://developer.android.com/guide/components/activities/ *Activity*-lifecycle?hl=id

Google. (2024). NestedScrollView. Diakses pada 7 November 2024, dari https://developer. android.com/reference/androidx/core/widget/NestedScrollView

Google. (2024). ScrollView. Diakses pada 3 November 2024, dari https://developer.android.com/ reference/android/widget/ScrollView?hl=id

Google. (2024). SearchView. Diakses pada 7 November 2024, dari https://developer. android.com/reference/android/widget/SearchView

Google. (2024). TabHost. Diakses pada 7 November 2024, dari https://developer. android.com/reference/android/widget/TabHost

Google. (2024). *WebView*. Diakses pada 7 November 2024, dari https://developer. android.com/reference/android/webkit/*WebView*

Prabowo, Iwan Ady, Wijayanto, Hendro, Yudanto, Bramasto Wiryawan, & Nugroho, Sapto. (2020). Pemrograman Mobile Berbasis Android (Teori, Latihan dan Tugas Mandiri). Lembaga Penelitian dan

DUMMY

Pengabdian Kepada Masyarakat Universitas Dian Nuswantoro Semarang. ISBN 978-623-95814-2-8.

Prasetyo, Novian Adi, Gustalika, Muhamad Azrino, Usman, Muhammad Lulu Latif, Wibowo, Fahrudin Mukti, & Fadhil, Muhammad Rizqan. (2023). Android Kotlin untuk Pemula. EUREKA MEDIA AKSARA. ISBN 978-623-151-891-0.

Saputra, Wanvy Aritha. (2020). Pemrograman Berbasis Objek: Pemrograman Mobile dengan Android Studio. Banjarmasin: Poliban Press. ISBN: 978-623-7694-18-2.

DUMMY

GLOSARIUM

Activity: Layar atau halaman antarmuka pengguna dalam aplikasi Android.

Adapter: Komponen yang menghubungkan data dengan tampilan UI seperti *RecyclerView*.

Android Studio: IDE resmi untuk mengembangkan aplikasi Android.

API (Application Programming Interface): Antarmuka yang memungkinkan aplikasi berinteraksi dengan layanan atau fungsi lain.

AVD (Android Virtual Device): Emulator perangkat Android untuk menguji aplikasi tanpa perangkat fisik.

BroadcastReceiver: Komponen untuk menerima notifikasi atau pesan dari sistem atau aplikasi lain.

CardView: Elemen UI yang menampilkan konten dalam bentuk kartu dengan bayangan dan sudut melengkung.

Class: Struktur dasar dalam pemrograman yang mendefinisikan objek, atribut, dan metode.

Constraint Layout: Layout fleksibel yang mengatur posisi elemen UI menggunakan constraints.

Coroutine: Fitur Kotlin untuk menjalankan operasi asinkron tanpa memblokir UI.

Data Binding: Fitur yang menghubungkan data langsung ke elemen UI dalam aplikasi.

Dialog: Jendela pop-up sementara untuk interaksi dengan pengguna.

Drawable: Sumber daya grafis seperti gambar dan ikon yang digunakan dalam aplikasi.

Emulator: Alat virtual yang digunakan untuk menjalankan dan menguji aplikasi Android.

External Storage: Penyimpanan eksternal pada perangkat, seperti kartu SD.

Feature: Fungsi atau layanan yang ditawarkan oleh aplikasi Android.

Fragment: Bagian dari antarmuka pengguna yang dapat digunakan kembali dalam *Activity*.

DUMMY

Gradle : Sistem *Build* otomatis untuk mengelola dependensi dan konfigurasi proyek Android.

Gravity: Properti yang menentukan posisi elemen UI dalam sebuah container.

IDE (Integrated Development Environment): Lingkungan terpadu untuk pengembangan aplikasi, seperti Android Studio.

Intent: Objek yang digunakan untuk memulai *Activity* atau mengirim pesan antar komponen aplikasi.

Internal Storage: Penyimpanan yang hanya dapat diakses oleh aplikasi yang menyimpannya.

JSON (JavaScript Object Notation): Format data yang digunakan untuk pertukaran data antara aplikasi dan server.

Kotlin: Bahasa pemrograman modern yang digunakan dalam pengembangan aplikasi Android.

Layout: Struktur antarmuka yang menentukan posisi elemen UI pada layar.

Lifecycle: Siklus hidup komponen Android seperti *Activity* atau *Fragment*, dari pembuatan hingga penghancuran.

Linear Layout: Layout yang menampilkan elemen UI dalam satu baris vertikal atau horizontal.

LiveData: Objek yang menyimpan data yang dapat diperbarui dan ditampilkan secara otomatis di UI.

Logcat: Alat debugging yang menampilkan log sistem dan aplikasi secara real-time.

Manifest: File XML yang berisi informasi penting tentang aplikasi, seperti *Permission* dan aktivitas utama.

Navigation Component: Alat untuk mengelola navigasi antar layar dalam aplikasi Android.

***RecyclerView*:** Komponen untuk menampilkan daftar panjang data dengan performa tinggi.

Relative Layout: Layout yang memungkinkan penempatan elemen UI relatif satu sama lain.

DUMMY

Retrofit: Library untuk membuat request HTTP dan mengakses API dengan mudah dalam aplikasi Android.

Room Database: Sistem penyimpanan data lokal yang lebih mudah digunakan dibandingkan dengan SQLite.

SDK (Software Development Kit): Kumpulan alat untuk mengembangkan aplikasi Android.

SearchView: Komponen UI untuk fungsi pencarian dalam aplikasi.

Service: Komponen yang menjalankan tugas di latar belakang tanpa antarmuka pengguna.

SharedPreferences: API untuk menyimpan data preferensi pengguna dalam bentuk key-value.

Snackbar: Notifikasi kecil yang muncul di bagian bawah layar dengan opsi tindakan.

Storage: Metode penyimpanan data dalam aplikasi Android, termasuk internal, eksternal, dan cloud.

Thread: Jalur eksekusi yang memungkinkan operasi berjalan secara paralel dalam aplikasi.

Toast: Notifikasi kecil yang muncul sementara di layar untuk menampilkan pesan singkat.

Variable: Penyimpanan data dalam kode yang digunakan untuk menyimpan nilai atau informasi.

View: Objek dasar dalam antarmuka pengguna, seperti tombol, teks, dan gambar.

ViewModel: Tempat menyimpan data UI yang tidak hilang saat terjadi perubahan orientasi aplikasi.

WebView: Komponen UI yang memungkinkan aplikasi menampilkan konten web.

XML (Extensible Markup Language): Bahasa markup yang digunakan untuk mendefinisikan layout dan *resource* lainnya dalam aplikasi Android.Ketik glosarium disini

DUMMY

INDEKS

DUMMY

DUMMY

TENTANG PENULIS

Ade Kurniawan, S.Pd., M.Pd.T lahir di Koto Ranah 29 Juni 1992. Lulus S1 dari Prodi Pendidikan Teknik Informatika dan Komputer Fakultas Teknik UNP tahun 2015. Kemudian melanjutkan Studi Pascasarjana di S2 Pendidikan Teknologi dan Kejuruan di UNP dan tamat pada tahun 2018. Penulis merupakan dosen Program Studi Informatika di departemen Teknik Elektronika Fakultas Teknik Universitas Negeri Padang. Sebelumnya pernah bekerja sebagai staff pengajar di Akademi Komunitas Pesisir Selatan (2016-2020) dan Programmer di Ar Risalah Padang (2021-2023). Penulis bergabung di Universitas Negeri Padang pada tahun 2022 dan saat ini aktif mengajar dan mengembangkan aplikasi android dan telah mengembangkan aplikasi yang digunakan untuk sarana di Nagari berupa SIPA Nagari dan aplikasi untuk pengendalian hama tanaman dengan nama E-Komoditas.

Jusmardi, S. Kom., M.Pd.T lahir di Kuala Enok (Indragiri Hilir, Provinsi Riau), 14 Juli 1978. Menamatkan SMA di SMU Nusantara Jambi tahun 1997. Menyelesaikan program S1 di STMIK Nurdin Hamzah Jambi, Jurusan Sistem Informatika tahun 2003. Tahun 2014 bergabung di Akademi Komunitas Negeri Pesisir Selatan sebagai Tenaga Pengajar pada Jurusan Manajemen Informatika sampai 2020. selain itu juga menjadi Tenaga Pengajar di STKIP Pesisir Selatan pada Jurusan TIK dari tahun 2016 sampai 2020. Tahun 2016 kembali melanjutkan Pendidikan S2 ke Pascasarjana FT UNP dengan Program Studi Pendidikan Teknologi Kejuruan, khususnya dalam Pendidikan Teknik Informatika. Saat ini sebagai salah satu Pendidik/dosen di Prodi Informatika, Fakultas Teknik Universitas Negeri Padang

FMIPA

Widya Darwin, S.Pd,, M.Pd.T Lahir di Painan, 09 Mei 1993. Menamatkan SMA di SMAN 1 Painan tahun 2011. Menyelesaikan program S1 di Universitas Putra Indonesia "YPTK" (UPI "YPTK") Fakultas Teknik, Jurusan Pendidikan Teknik Informatika dan Komputer tahun 2015. Mulai mengabdikan diri sebagai guru honorer tahun 2014 di SMK Negeri 1 Painan, Tahun 2015 bergabung dengan AKN Pesisir Selatan sebagai tenaga pengajar. Tahun 2016 kembali melanjutkan ke Pascasarjana FT UNP dengan Program Studi Tendidikan Teknologi Kejuruan dan telah selesai pada Agustus 2020. Saat ini Penulis merupakan Dosen Program Studi Informatika di Departemen Teknik Elektronika Fakultas Teknik Universitas Negeri Padang.

Melri Deswina, S,Pd., M.Pd.T lahir di Salido, 17 Mei 1990. Lulus S1 dari Pendidikan Teknik Informatika dan Komputer Fakultas Teknik Universitas Negeri Padang (UNP) dan tamat pada tahun 2012. Mulai mengabdikan diri pada tahun 2012 di MTsN Salido Sebagai Operator Keuangan dan juga sebagai pegawai honorer di SMK N 1 Painan mengajar jurusan Teknik Komputer Jaringan (TKJ). Pada tahun 2014 bergabung dengan Akademi Komunitas Pesisir Selatan (AKN Pessel). Pada tahun 2015 melanjutkan Kembali di Pascasarjana FT UNP dengan jurusan Pendidikan Teknologi dan Kejuruan dan tamat pada tahun 2017. Saat ini penulis merupakan dosen Program Studi Informatika di Departemen Teknik Elektronika Fakultas Teknik Universitas Negeri Padang dan juga bergabung sebagai dosen tutor di Universitas Terbuka.

DRAFT

RINGKASAN ISI BUKU

Buku Pemrograman Mobile Android dengan Kotlin ini dirancang untuk membantu mahasiswa atau pemula untuk untuk memahami dasar hingga lanjutan pengembangan aplikasi Android menggunakan Bahasa pemrograman Kotlin. Buku ini memberikan panduan komprehensif tentang bagaimana membangun aplikasi yang efektif, modern, dan sesuai dengan standar pengembangan Android terkini.

Buku ini disusun secara sistematis, dimulai dari pengenalan Kotlin sebagai bahasa pemrograman untuk Android hingga membangun aplikasi yang lengkap dengan fitur-fitur seperti dialog, API, notifikasi, adapter dan lainnya. Setiap bab dilengkapi dengan teori, langkah-langkah implementasi, contoh kode, dan tugas praktikum untuk memperkuat pemahaman pembaca.

DUMMY