

LAPORAN PROJECT TUGAS PPK 2025
MATA KULIAH PEMROGAMAN PLATFORM KHUSUS
APLIKASI BMI & KESEHATAN BERBASIS ANDROID



Dosen Pembimbing :
Ibnu Santoso, S.S.T., M.T.

Disusun Oleh :
Ananda Mizan Ali (222312970)
Kelas 3SI2

PROGRAM STUDI KOMPUTASI STATISTIK
POLITEKNIK STATISTIKA STIS
JAKARTA
2025

DAFTAR ISI

BAB I.....	1
PENDAHULUAN	1
A. Latar Belakang.....	1
B. Tujuan	1
BAB II.....	2
METODOLOGI DAN PERANCANGAN SISTEM.....	2
A. Platform dan Teknologi.....	2
B. Struktur Aplikasi	2
C. Alur Kerja Aplikasi.....	3
BAB III	4
IMPLEMENTASI SISTEM.....	4
A. Model Data.....	4
B. Logika Algoritma dan Perhitungan	5
BAB IV	10
HASIL DAN PEMBAHASAN.....	10
A. Tampilan Antarmuka Aplikasi	10
B. Pengujian Fungsional.....	13
C. Pembahasan Implementasi Kode	17
D. Analisis Kelebihan dan Kekurangan.....	17
BAB V	18
PENUTUP.....	18
A. Kesimpulan	18
B. Daftar Pustaka.....	18
LAMPIRAN.....	20

BAB I

PENDAHULUAN

A. Latar Belakang

Kesehatan tubuh merupakan aspek penting yang perlu diperhatikan dalam kehidupan sehari-hari. Salah satu indikator kesehatan yang sering digunakan adalah Body Mass Index (BMI) untuk mengetahui status berat badan seseorang. Selain itu, faktor lain seperti kadar kolesterol, asam urat, dan gula darah juga berpengaruh terhadap kondisi kesehatan. Oleh karena itu, dikembangkan aplikasi Android sederhana yang dapat membantu pengguna menghitung BMI serta memberikan evaluasi kesehatan dan rekomendasi diet secara mandiri.

B. Tujuan

Tujuan pembuatan aplikasi ini adalah:

1. Menghitung nilai BMI berdasarkan data tinggi dan berat badan.
2. Menampilkan kategori BMI pengguna.
3. Memberikan evaluasi kondisi kesehatan sederhana.
4. Menyajikan rekomendasi rencana diet sesuai kategori BMI.

BAB II

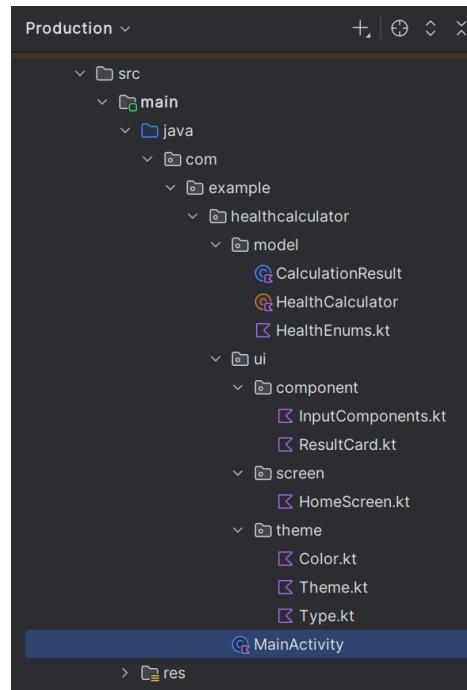
METODOLOGI DAN PERANCANGAN SISTEM

A. Platform dan Teknologi

Aplikasi ini dikembangkan menggunakan:

- Bahasa Pemrograman: Kotlin (Versi terbaru).
- Framework UI: Jetpack Compose (Declarative UI Toolkit).
- Desain UI: Material Design 3 (M3) dengan dukungan tema dinamis.
- Arsitektur: *Single Activity Architecture* dengan pemisahan *Model* dan *View*.
- Minimum SDK: Android 7.0 (Nougat) / API Level 24.
- Koneksi: Offline (Semua pemrosesan data dilakukan secara lokal di perangkat).

B. Struktur Aplikasi



1. com.example.healthcalculator

Main Package yang berisi MainActivity.kt sebagai titik masuk (*entry point*) aplikasi.

2. model (Logika & Data)

- Berisi logika bisnis utama dan representasi data.
- Menangani perhitungan BMI dan validasi parameter kesehatan berdasarkan umur dan jenis kelamin.
- Menyimpan *Enum* untuk kategori (Gender, AgeCategory, SugarTestType).

3. ui (Antarmuka Pengguna)

Terbagi menjadi tiga sub-package:

- **screen:** Berisi halaman utama (HomeScreen.kt) yang menyatukan seluruh elemen tampilan.
- **components:** Berisi komponen UI yang dapat digunakan kembali (*reusable*), seperti *Input Form* (HealthInputRow, SugarTypeDropdown) dan *Result Card*.
- **theme:** Mengatur konfigurasi visual seperti palet warna (*Color Palette*) untuk Light/Dark Mode dan tipografi.

C. Alur Kerja Aplikasi

Alur kerja sistem dirancang untuk menangani input multi-variabel guna menghasilkan analisis kesehatan yang akurat:

1. Input Data Demografi

Pengguna memasukkan data pribadi yang mempengaruhi standar nilai normal, yaitu:

- Umur (untuk menentukan kategori Anak/Remaja/Dewasa/Lansia).
- Jenis Kelamin (khusus validasi Asam Urat).
- Tinggi Badan dan Berat Badan.

2. Input Data Klinis

Pengguna memasukkan hasil tes laboratorium dengan spesifikasi:

- Tipe Tes Gula Darah (Puasa / Sebelum Makan / 2 Jam Setelah Makan).
- Nilai Gula Darah, Kolesterol Total, dan Asam Urat.

3. Validasi Input

Sistem memeriksa kelengkapan data dan memastikan input berupa angka valid. Jika data tidak lengkap, sistem menampilkan pesan peringatan (Snackbar).

4. Pemrosesan Logika (Calculation)

Sistem melakukan perhitungan secara bertingkat:

- Menghitung BMI dan menentukan kategori berat badan.
- Mencocokkan nilai Gula Darah berdasarkan Umur dan Tipe Tes.
- Mencocokkan nilai Asam Urat berdasarkan Umur dan Jenis Kelamin.
- Mencocokkan nilai Kolesterol berdasarkan Umur.

5. Output Hasil

Aplikasi menampilkan:

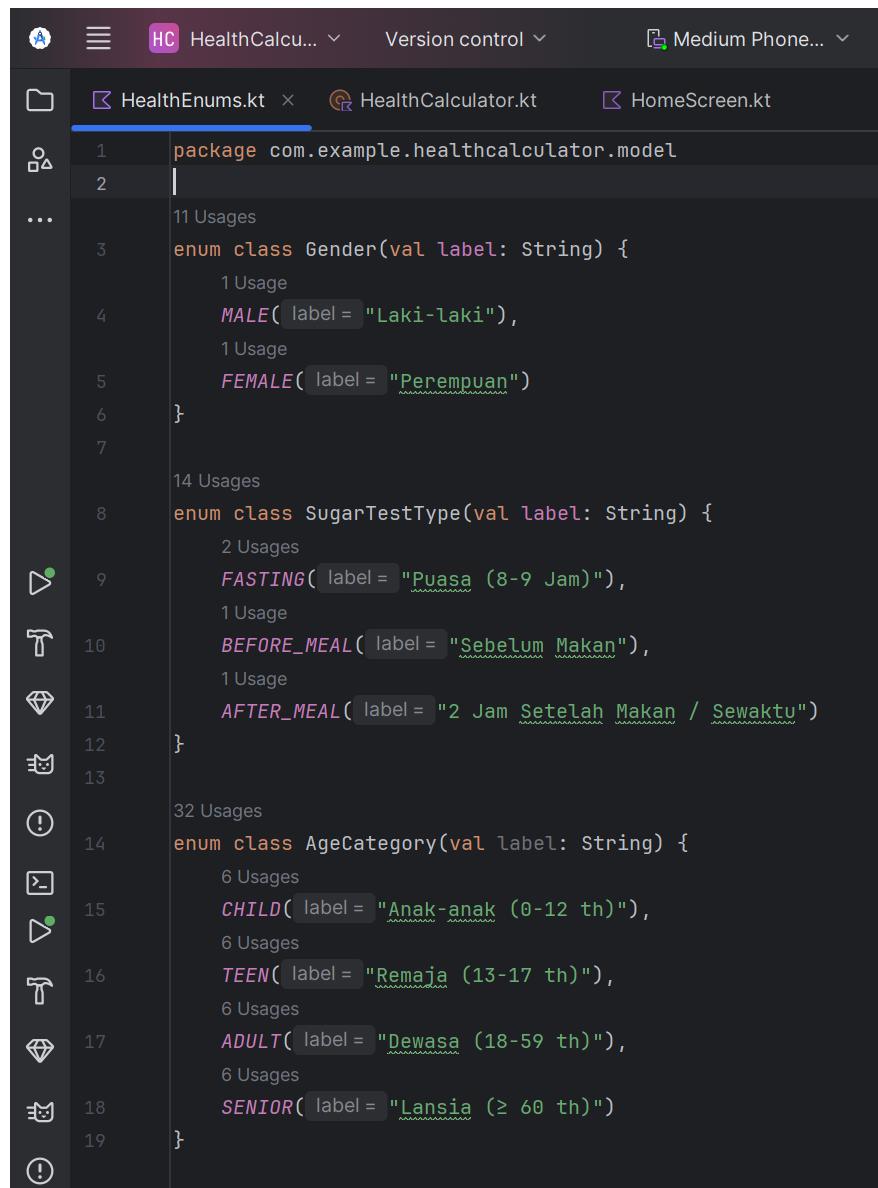
- Nilai BMI beserta kategori warnanya.
- Daftar peringatan kesehatan jika ada parameter darah yang melebihi batas normal.
- Rekomendasi kalori dan menu diet yang disesuaikan dengan kategori BMI pengguna.

BAB III

IMPLEMENTASI SISTEM

A. Model Data

Model data atau domain model dijelaskan dalam file HealthEnums.kt yang mendefinisikan opsi tetap (konstanta) untuk aplikasi. Secara spesifik, file ini dipakai untuk standarisasi data, keamanan tipe (type safety), dan kemudahan logika perhitungan di aplikasi Android (Jetpack Compose).



```
package com.example.healthcalculator.model

enum class Gender(val label: String) {
    MALE(label = "Laki-laki"),
    FEMALE(label = "Perempuan")
}

enum class SugarTestType(val label: String) {
    FASTING(label = "Puasa (8-9 Jam)"),
    BEFORE_MEAL(label = "Sebelum Makan"),
    AFTER_MEAL(label = "2 Jam Setelah Makan / Sewaktu")
}

enum class AgeCategory(val label: String) {
    CHILD(label = "Anak-anak (0-12 th)"),
    TEEN(label = "Remaja (13-17 th)"),
    ADULT(label = "Dewasa (18-59 th)"),
    SENIOR(label = "Lansia (> 60 th)")
}
```

B. Logika Algoritma dan Perhitungan

Proses pengolahan data dalam aplikasi ini terpusat pada berkas HealthCalculator.kt. Perancangan aplikasi menerapkan prinsip Separation of Concerns, di mana logika bisnis dan proses perhitungan dipisahkan secara jelas dari lapisan antarmuka pengguna yang dibangun menggunakan Jetpack Compose. Seluruh fungsi perhitungan dirancang secara deterministik, sehingga setiap kombinasi input yang sama akan selalu menghasilkan keluaran yang konsisten dan tidak dipengaruhi oleh kondisi atau status eksternal aplikasi.

Berikut adalah rincian algoritma untuk setiap parameter kesehatan:

1. Kategorisasi Usia (Age Categorization)

Sebelum melakukan validasi medis, sistem terlebih dahulu mengelompokkan usia pengguna ke dalam AgeCategory menggunakan struktur kontrol when. Hal ini krusial karena setiap kelompok usia memiliki ambang batas normal yang berbeda.

- **Aturan:**

- 0–12 tahun: Anak-anak
- 13–17 tahun: Remaja
- 18–59 tahun: Dewasa
- ≥ 60 tahun: Lansia

- **Logika Code :**

```
1 Usage
fun getAgeCategory(age: Int): AgeCategory {
    return when (age) {
        in 0 .. 12 -> AgeCategory.CHILD
        in 13 .. 17 -> AgeCategory.TEEN
        in 18 .. 59 -> AgeCategory.ADULT
        else -> AgeCategory.SENIOR
    }
}
```

2. Perhitungan Body Mass Index (BMI)

Perhitungan BMI digunakan untuk menentukan status berat badan pengguna dan menjadi acuan utama dalam memberikan rekomendasi diet (kalori).

- **Rumus :**

$$\text{BMI} = \frac{\text{Berat Badan}}{(\text{Tinggi Badan})^2}$$

- **Algoritma :**

- a. Konversi tinggi badan dari cm ke meter (cm / 100).

- b. Hitung nilai BMI.
- c. Klasifikasikan hasil berdasarkan standar WHO/Kemenkes:
 - o < 18.5 : **Kurus (Underweight)**
 - o 18.5 - 24.9 : **Normal**
 - o 25.0 - 29.9 : **Overweight**
 - o > 30.0 : **Obesitas**

- **Logika Code :**

```
// 1. Hitung BMI
val heightM = heightCm / 100.0
val bmi = weightKg / (heightM * heightM)
val bmiCategory = when {
    bmi < 18.5 -> "Kurus"
    bmi < 25.0 -> "Normal"
    bmi < 30.0 -> "Overweight"
    else -> "Obesitas"
}
```

3. Evaluasi Gula Darah

Evaluasi gula darah memiliki kompleksitas logika paling tinggi karena bergantung pada dua variabel input sekaligus: **Kategori Usia** dan **Tipe Tes** (Puasa, Sebelum Makan, atau Setelah Makan).

- **Mekanisme Validasi:**

Menggunakan Nested When Expression (Percabangan bertingkat).

- a. **Cek Keselamatan** : Jika nilai < 60 mg/dL, sistem langsung mengembalikan status "**Bahaya : Hipoglikemia**" untuk semua umur.
- b. **Penentuan Range Normal :**

- **Puasa (Fasting):**

- o Anak/Remaja: 70–100 mg/dL
- o Dewasa: 70–99 mg/dL
- o Lansia: 70–110 mg/dL

- **Sebelum Makan:**

- o Non-Lansia: 70–130 mg/dL
- o Lansia: 80–140 mg/dL

- **2 Jam Setelah Makan:**

- o Anak/Remaja: Max 140 mg/dL
- o Dewasa/Lansia: Max 200 mg/dL

- **Output :**

Jika nilai berada di luar *Range* yang ditentukan, sistem memberikan label "Rendah" atau "Tinggi".

- **Logika Code :**

```
private fun evaluateBloodSugar(category: AgeCategory, type: SugarTestType, value: Double): String {
    // Batas bawah aman mutlak
    if (value < 60) return "Bahaya: Hipoglikemia (< 60)"
    val range = when (type) {
        SugarTestType.FASTING -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN -> 70.0 .. ≤ 100.0
            AgeCategory.ADULT -> 70.0 .. ≤ 99.0
            AgeCategory.SENIOR -> 70.0 .. ≤ 110.0
        }
        SugarTestType.BEFORE_MEAL -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN, AgeCategory.ADULT -> 70.0 .. ≤ 130.0
            AgeCategory.SENIOR -> 80.0 .. ≤ 140.0
        }
        SugarTestType.AFTER_MEAL -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN -> 0.0 .. ≤ 139.9 // < 140
            AgeCategory.ADULT, AgeCategory.SENIOR -> 0.0 .. ≤ 199.9 // < 200
        }
    }
    return when {
        value < range.start -> "Rendah (< ${range.start})"
        value > range.endInclusive -> "Tinggi (> ${range.endInclusive.toInt()})"
        else -> "Normal"
    }
}
```

4. Evaluasi Kolesterol Total

Validasi kolesterol total disederhanakan berdasarkan batas atas aman yang berbeda antara kelompok usia muda dan dewasa.

- **Aturan:**

- **Anak & Remaja:** Batas aman < 170 mg/dL.
- **Dewasa & Lansia:** Batas aman < 200 mg/dL.

- **Logika Code:**

Sistem mengecek apakah input pengguna melebihi batas atas (maxLimit). Jika ya, status menjadi "Tinggi".

```
private fun evaluateCholesterol(category: AgeCategory, value: Double): String {
    // Range Kadar Kolesterol
    val maxLimit = when (category) {
        AgeCategory.CHILD, AgeCategory.TEEN -> 170.0
        AgeCategory.ADULT, AgeCategory.SENIOR -> 200.0
    }
    return if (value > maxLimit) "Tinggi (> ${maxLimit.toInt()})" else "Normal"
}
```

5. Evaluasi Asam Urat (Uric Acid)

Berbeda dengan parameter lain, evaluasi asam urat pada orang dewasa dipengaruhi oleh **Jenis Kelamin (Gender)** secara biologis.

- **Algoritma Penentuan Range:**

- a. **Anak (0-12):** Range 2.0 – 5.5 mg/dL.
- b. **Remaja (13-17):** Range 2.5 – 6.5 mg/dL.
- c. **Dewasa & Lansia:**
 - o Jika **Perempuan:** Range 2.4 – 6.0 mg/dL.
 - o Jika **Laki-laki:** Range 3.4 – 7.0 mg/dL.

- **Logika Code :**

```
private fun evaluateUricAcid(category: AgeCategory, gender: Gender, value: Double): String {  
    val range = when (category) {  
        AgeCategory.CHILD -> 2.0 .. 5.5  
        AgeCategory.TEEN -> 2.5 .. 6.5  
        AgeCategory.ADULT, AgeCategory.SENIOR -> if (gender == Gender.FEMALE)  
            2.4 .. 6.0 else 3.4 .. 7.0  
    }  
    return when {  
        value < range.start -> "Rendah (< ${range.start})"  
        value > range.endInclusive -> "Tinggi (> ${range.endInclusive})"  
        else -> "Normal"  
    }  
}
```

6. Sistem Evaluasi dan Rekomendasi Diet

Sistem rekomendasi tidak menggunakan logika kondisional yang rumit, melainkan pemetaan langsung (*Direct Mapping*) dari **Kategori BMI**.

- **Mapping :**

- a. **Kurus:** Target ± 2500 kkal (Surplus kalori) -> Menu tinggi protein & karbohidrat.
- b. **Normal:** Target ± 2200 kkal (Maintenance) -> Menu seimbang.
- c. **Overweight:** Target ± 1800 kkal (Defisit ringan) -> Kurangi karbohidrat sederhana.
- d. **Obesitas:** Target ± 1500 kkal (Defisit ketat) -> Fokus serat dan protein rendah lemak.

- **Logika Code :**

```
// Hasil Evaluasi
val warnings = mutableListOf<String>()
if (sugarStatus != "Normal") warnings.add("Gula Darah: $sugarStatus")
if (cholesterolStatus != "Normal") warnings.add("Kolesterol: $cholesterolStatus")
if (uricStatus != "Normal") warnings.add("Asam Urat: $uricStatus")
if (warnings.isEmpty()) warnings.add("Semua Parameter Normal ✓")

// Rekomendasi Diet
val (calories, menu) = when (bmiCategory) {
    "Kurus" -> "+2500 kkal/hari" to "Nasi, ayam, telur, susu, buah"
    "Normal" -> "+2200 kkal/hari" to "Nasi merah, sayur, ikan, buah"
    "Overweight" -> "+1800 kkal/hari" to "Sayur, dada ayam, oatmeal"
    else -> "+1500 kkal/hari" to "Sayur rebus, buah, protein rendah lemak"
}
```

BAB IV

HASIL DAN PEMBAHASAN

A. Tampilan Antarmuka Aplikasi

Bagian ini menyajikan hasil implementasi antarmuka pengguna (*User Interface*) yang dibangun menggunakan Jetpack Compose dengan gaya desain Material Design 3.

1. Halaman Utama (Input Data)

Aplikasi menampilkan formulir input yang responsif. Pengguna diminta memasukkan data demografi (Umur, Gender, Tinggi, Berat) dan data klinis (Tipe Tes Gula, Nilai Gula, Kolesterol, Asam Urat).

The screenshot shows the main screen of the "Health & Diet Planner" application. At the top, there is a green header bar with the title "Health & Diet Planner". Below the header, the text "Data Pribadi" is displayed. There is a text input field labeled "Umur (Tahun)". Below it, there is a section for "Jenis Kelamin:" with two radio buttons: one for "Laki-laki" and one for "Perempuan". To the left of the gender buttons is a text input field labeled "Tinggi (cm)". To the right of the gender buttons is another text input field labeled "Berat (kg)". In the "Hasil Lab" section, there are four text input fields: "Tipe Tes Gula Darah:" (with a dropdown menu showing "Puasa (8-9 Jam)"), "Gula Darah (mg/dL)", "Kolesterol Total (mg/dL)", and "Asam Urat (mg/dL)". At the bottom, there is a large green button labeled "Analisa Kesehatan".

2. Tampilan Hasil Analisa (Output)

Setelah tombol "Analisa Kesehatan" ditekan, hasil perhitungan ditampilkan dalam bentuk Card yang informatif. Warna indikator kategori BMI berubah secara dinamis (Biru/Hijau/Oranye/Merah) sesuai tingkat risiko.

The screenshot shows the 'Health & Diet Planner' app interface. At the top, there's a header bar with the time (9:18), battery level, and signal strength. Below it is a green header 'Health & Diet Planner'. The main area has two sections: 'Data Pribadi' and 'Hasil Lab'. In 'Data Pribadi', users enter their age (24), gender (Laki-laki selected), height (180 cm), and weight (60 kg). In 'Hasil Lab', users select their blood test type (post-meal), and enter blood sugar (175 mg/dL), total cholesterol (160 mg/dL), and uric acid (4.3 mg/dL). A large green button at the bottom is labeled 'Analisa Kesehatan'.

9:18 ⚡

Health & Diet Planner

Data Pribadi

Umur (Tahun) —————
24

Jenis Kelamin:

Laki-laki Perempuan

Tinggi (cm) —————
180

Berat (kg) —————
60

Hasil Lab

Tipe Tes Gula Darah:

2 Jam Setelah Makan / Sewaktu

Gula Darah (mg/dL) —————
175

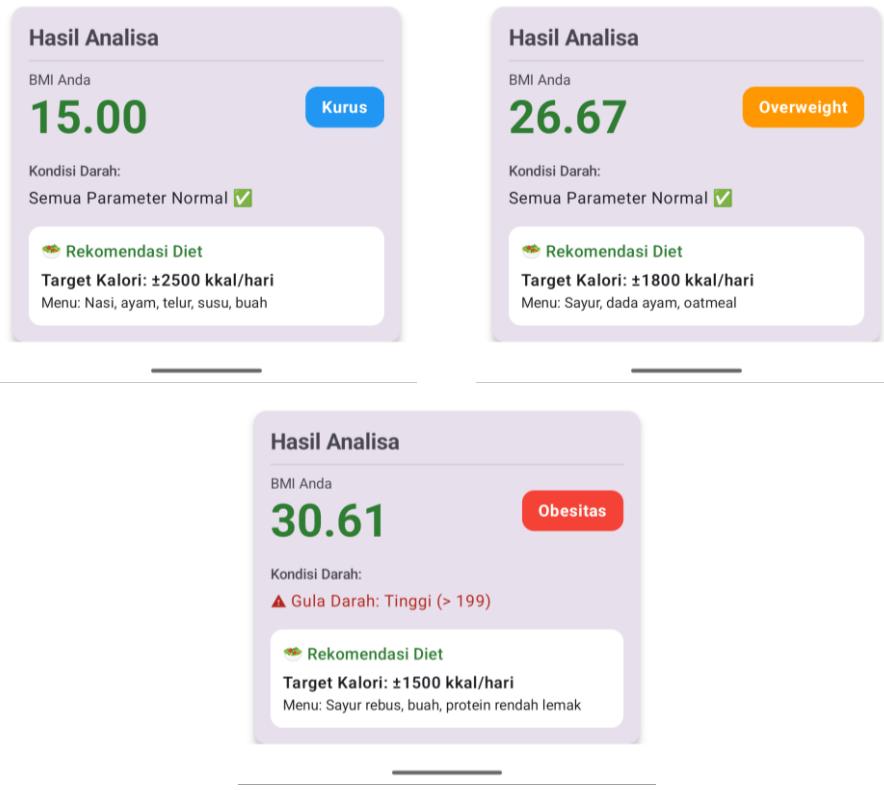
Kolesterol Total (mg/dL) —————
160

Asam Urat (mg/dL) —————
4.3

Analisa Kesehatan

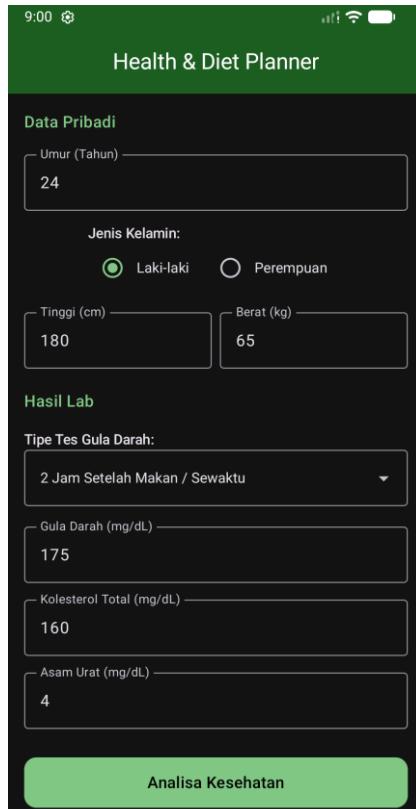


Tampilan Hasil Analisis Lain berdasarkan BMI :



3. Implementasi Dark Mode

Aplikasi mendukung fitur tema gelap (Dark Mode) yang otomatis menyesuaikan dengan pengaturan sistem perangkat untuk kenyamanan mata pengguna.

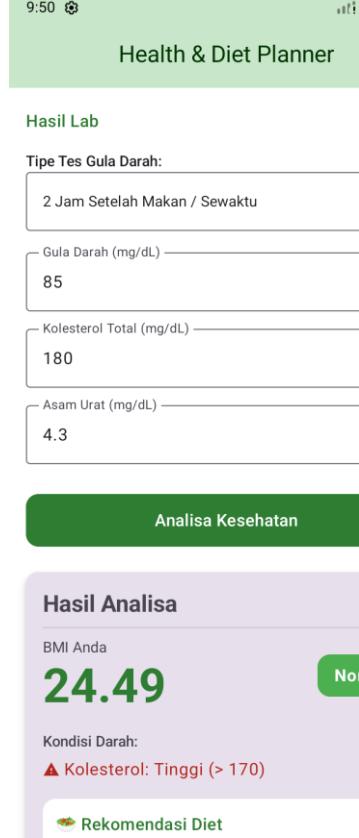




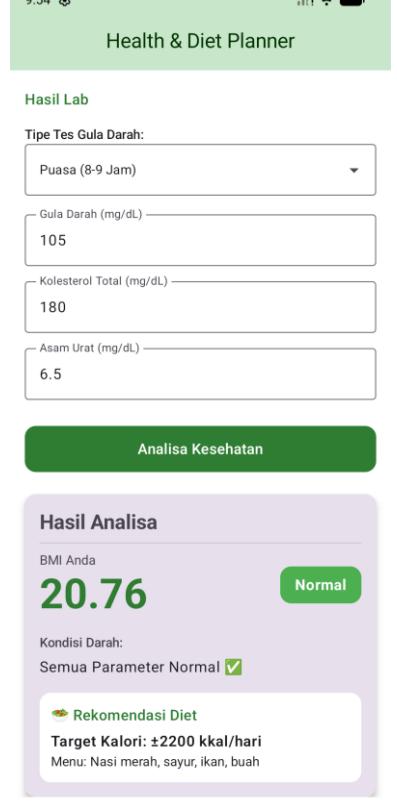
B. Pengujian Fungsional

1. Pengujian Logika Kompleks (Umur, Gender, & Tipe Tes)

Tujuannya adalah memastikan validasi medis berjalan sesuai tabel standar yang rumit (beda umur/gender = beda batas normal).

No.	Fokus Pengujian	Data Input	Hasil
1	Validasi Anak-Anak (Batas Kolesterol Lebih Ketat)	<ul style="list-style-type: none"> Umur: 10 Th (Anak) Kolesterol: 180 mg/dL (Harusnya Tinggi bagi anak) Yang lain normal Gula darah : 85 mg/dL Asam urat : 3 mg/dL 	 <p>9:50 ⓘ</p> <p>Health & Diet Planner</p> <p>Hasil Lab</p> <p>Tipe Tes Gula Darah:</p> <p>2 Jam Setelah Makan / Sewaktu</p> <p>Gula Darah (mg/dL) — 85</p> <p>Kolesterol Total (mg/dL) — 180</p> <p>Asam Urat (mg/dL) — 4.3</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p> <p>BMI Anda</p> <p>24.49 Normal</p> <p>Kondisi Darah:</p> <p>▲ Kolesterol: Tinggi (> 170)</p> <p>Rekomendasi Diet</p> <p>Target Kalori: ±2200 kkal/hari</p> <p>Menu: Nasi merah, sayur, ikan, buah</p>

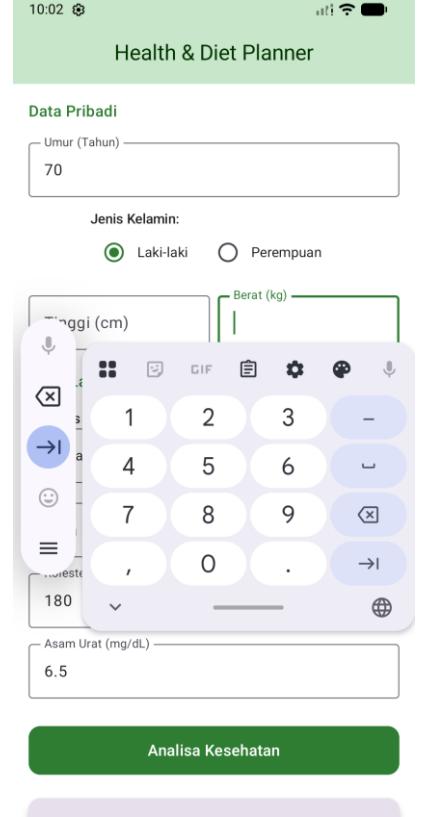
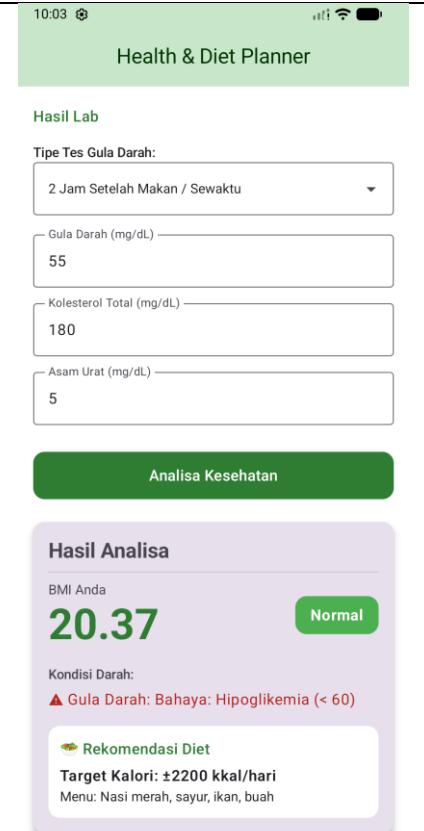
2	Validasi Gender (Asam Urat Wanita)	<ul style="list-style-type: none"> Umur: 30 Th (Dewasa) Gender: Perempuan Asam Urat: 6.5 mg/dL Batas Pria: 7.0 Batas Wanita: 6.0 Input 6.5 Tinggi bagi wanita (tapi normal bagi pria). 	<p>9:52 ⓘ ⚡</p> <p>Health & Diet Planner</p> <p>Hasil Lab</p> <p>Tipe Tes Gula Darah:</p> <p>2 Jam Setelah Makan / Sewaktu</p> <p>Gula Darah (mg/dL) _____ 85</p> <p>Kolesterol Total (mg/dL) _____ 180</p> <p>Asam Urat (mg/dL) _____ 6.5</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p> <p>BMI Anda 26.67 Overweight</p> <p>Kondisi Darah: ▲ Asam Urat: Tinggi (> 6.0)</p> <p>Rekomendasi Diet Target Kalori: ±1800 kkal/hari Menu: Sayur, dada ayam, oatmeal</p>
3	Validasi Tipe Tes Gula (Puasa vs Sewaktu)	<ul style="list-style-type: none"> Umur: 25 Th Tipe: Puasa Gula: 110 mg/dL Batas Puasa: 100 Batas Sewaktu: 200 Input 110 Tinggi untuk puasa. 	<p>9:53 ⓘ ⚡</p> <p>Health & Diet Planner</p> <p>Hasil Lab</p> <p>Tipe Tes Gula Darah:</p> <p>Puasa (8-9 Jam)</p> <p>Gula Darah (mg/dL) _____ 110</p> <p>Kolesterol Total (mg/dL) _____ 180</p> <p>Asam Urat (mg/dL) _____ 6.5</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p> <p>BMI Anda 20.76 Normal</p> <p>Kondisi Darah: ▲ Gula Darah: Tinggi (> 99)</p> <p>Rekomendasi Diet Target Kalori: ±2200 kkal/hari Menu: Nasi merah, sayur, ikan, buah</p>

4	Validasi Lansia (Toleransi Gula Lebih Tinggi)	<ul style="list-style-type: none"> Umur: 70 Th (Lansia) Tipe: Puasa Gula: 105 mg/dL Batas Dewasa: 99 Batas Lansia: 110 Input 105 Normal bagi Lansia. 	 <p>Hasil Lab</p> <p>Tipe Tes Gula Darah:</p> <p>Puasa (8-9 Jam)</p> <p>Gula Darah (mg/dL) 105</p> <p>Kolesterol Total (mg/dL) 180</p> <p>Asam Urat (mg/dL) 6.5</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p> <p>BMI Anda 20.76 Normal</p> <p>Kondisi Darah: Semua Parameter Normal ✓</p> <p>Rekomendasi Diet</p> <p>Target Kalori: ±2200 kcal/hari</p> <p>Menu: Nasi merah, sayur, ikan, buah</p>
---	---	--	--

2. Pengujian Validasi Input

Tujuannya adalah memastikan aplikasi tidak *crash* jika user melakukan kesalahan input.

No.	Skenario	Tindakan Pengguna	Respon Sistem	Hasil
1	Input Kosong	Mengosongkan salah satu atau semua kolom, lalu tekan tombol "Analisa".	<ul style="list-style-type: none"> Keyboard tertutup otomatis. Muncul Snackbar: "Mohon lengkapi semua data..." Tidak ada perhitungan terjadi. 	 <p>10:01 ⚡ Health & Diet Planner</p> <p>Data Pribadi</p> <p>Umur (Tahun) 70</p> <p>Jenis Kelamin:</p> <p><input checked="" type="radio"/> Laki-laki <input type="radio"/> Perempuan</p> <p>Tinggi (cm) Berat (kg)</p> <p>Hasil Lab</p> <p>Tipe Tes Gula Darah:</p> <p>Puasa (8-9 Jam)</p> <p>Gula Darah (mg/dL) 105</p> <p>Kolesterol Total (mg/dL) 180</p> <p>Asam Urat (mg/dL) 6.5</p> <p>Analisa Kesehatan</p> <p>Mohon lengkapi semua data dengan benar.</p>

2	Input Karakter Illegal Mencoba mengetik huruf "A" atau simbol "@" pada kolom berat badan.	<ul style="list-style-type: none"> Kolom input menolak karakter tersebut. Hanya angka dan titik (.) yang muncul. 	 <p>10:02 ④ Health & Diet Planner</p> <p>Data Pribadi</p> <p>Umur (Tahun) — 70</p> <p>Jenis Kelamin:</p> <p><input checked="" type="radio"/> Laki-laki <input type="radio"/> Perempuan</p> <p>Tinggi (cm) Berat (kg)</p> <p>Asam Urat (mg/dL) — 6.5</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p>
3	Input Gula Darah Bahaya Memasukkan Gula Darah 40 mg/dL (Sangat rendah).	<ul style="list-style-type: none"> Muncul Warning Spesifik: ⚠️ Bahaya: Hipoglikemia (< 60) 	 <p>10:03 ④ Health & Diet Planner</p> <p>Hasil Lab</p> <p>Tipe Tes Gula Darah: 2 Jam Setelah Makan / Sewaktu</p> <p>Gula Darah (mg/dL) — 55</p> <p>Kolesterol Total (mg/dL) — 180</p> <p>Asam Urat (mg/dL) — 5</p> <p>Analisa Kesehatan</p> <p>Hasil Analisa</p> <p>BMI Anda 20.37 Normal</p> <p>Kondisi Darah: ⚠️ Gula Darah: Bahaya: Hipoglikemia (< 60)</p> <p>Rekomendasi Diet</p> <p>Target Kalori: ±2200 kkal/hari Menu: Nasi merah, sayur, ikan, buah</p>

C. Pembahasan Implementasi Kode

Pada tahap ini dibahas mengenai bagaimana logika kode bekerja menangani kompleksitas aturan medis.

1. Logika Penentuan Kategori Usia

Aplikasi tidak menggunakan batasan umur statis, melainkan mengelompokkan umur terlebih dahulu menggunakan Enum Class AgeCategory. Hal ini memudahkan validasi karena setiap kategori (Anak, Remaja, Dewasa, Lansia) memiliki ambang batas berbeda. Implementasi ini terdapat pada file HealthCalculator.kt.

2. Penanganan Logika Bersarang (Nested Logic)

Untuk parameter Gula Darah, sistem menerapkan logika bersarang (Nested When) karena validasi bergantung pada dua variabel input: Kategori Usia dan Tipe Tes.

Contoh :

Jika pengguna memilih tipe tes "Puasa", batas normal anak-anak adalah 100 mg/dL, sedangkan jika memilih "Sewaktu", batasnya naik menjadi 140 mg/dL. Kode program berhasil menangani variasi ini secara dinamis.

3. Penerapan Jetpack Compose

Penggunaan Jetpack Compose memungkinkan UI dibangun secara deklaratif. Komponen seperti LazyColumn digunakan agar tampilan dapat digulir (scrollable) pada perangkat layar kecil, memastikan tidak ada elemen UI yang tertutup keyboard.

D. Analisis Kelebihan dan Kekurangan

Berdasarkan hasil pengujian, berikut adalah analisis performa aplikasi:

Kelebihan:

- Akurasi Medis :** Perhitungan sudah membedakan standar antara Anak, Dewasa, dan Lansia, serta membedakan Gender untuk asam urat, sehingga hasil lebih valid secara medis.
- Efisiensi :** Aplikasi berjalan luring (*offline*) tanpa *loading* data dari internet, sehingga proses perhitungan instan.
- User Experience (UX):** Validasi input mencegah pengguna memasukkan huruf atau membiarkan form kosong, serta dukungan Dark Mode meningkatkan kenyamanan.

Kekurangan (Saran Pengembangan):

- Belum adanya fitur penyimpanan riwayat (History) karena batasan tidak menggunakan database.
- Belum ada fitur ekspor hasil ke PDF untuk keperluan medis.

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian sistem yang telah dilakukan pada aplikasi **Health Calculator**, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Fungsionalitas Aplikasi

Aplikasi berhasil dibangun menggunakan bahasa pemrograman Kotlin dan framework Jetpack Compose dengan arsitektur Single Activity. Seluruh fitur utama, mulai dari input data demografi, perhitungan BMI, hingga validasi parameter kesehatan, berjalan dengan baik tanpa kendala (bug) yang signifikan.

2. Akurasi Logika Medis

Sistem mampu menerapkan logika validasi kesehatan yang kompleks sesuai standar medis. Aplikasi tidak hanya menghitung angka statis, tetapi berhasil membedakan standar nilai normal berdasarkan variabel Kategori Usia (Anak, Remaja, Dewasa, Lansia), Jenis Kelamin, dan Tipe Tes (Puasa/Sewaktu).

3. Kemandirian Sistem (Offline)

Aplikasi beroperasi sepenuhnya secara luring (offline) tanpa memerlukan koneksi internet atau web service. Hal ini menjamin privasi data pengguna karena seluruh pemrosesan dilakukan secara lokal di perangkat pengguna.

4. Antarmuka Pengguna (UI/UX)

Implementasi Material Design 3 memberikan tampilan yang modern dan intuitif. Dukungan fitur Dark Mode yang otomatis mengikuti pengaturan sistem telah berhasil diterapkan untuk meningkatkan kenyamanan visual pengguna.

5. Manfaat Praktis

Aplikasi ini memberikan nilai tambah berupa rekomendasi rencana diet (target kalori dan menu makanan) yang dipersonalisasi berdasarkan kategori BMI pengguna, sehingga dapat menjadi alat bantu pemantauan kesehatan harian yang efektif.

B. Daftar Pustaka

4. Referensi Teknis (Android Development)

Google Developers. (2025). *Build better apps faster with Jetpack Compose*. Android Developers Documentation. Diakses dari <https://developer.android.com/develop/ui/compose/documentation?hl=id>

Google Developers. (2025). *Material Design 3 in Compose*. Android Developers Documentation. Diakses dari
<https://developer.android.com/develop/ui/compose/designsystems/material3?hl=id>

5. Referensi Medis (Standar Kesehatan)

Alodokter. (2024). *Indeks massa tubuh, cara mengetahui dan 4 kategorinya*. Diakses dari <https://www.alodokter.com/pemahaman-seputar-indeks-massa-tubuh>

Alodokter. (2016). *Kadar normal gula darah, kolesterol, dan asam urat*. Diakses dari <https://www.alodokter.com/komunitas/topic/berapa-normal-gula-darah-kolesterol-dan-asam-urat>

Alodokter. (2019). *Berapakah nilai normal gula darah, asam urat dan kolesterol pada perempuan dan laki-laki*. Diakses dari <https://www.alodokter.com/komunitas/topic/angka-normal-sewaktu-gula-darah-kolesterol-asam-urat-pada-perempuan-dan-laki2>

Alodokter. (2021). *Nilai kadar normal*. Diakses dari <https://www.alodokter.com/komunitas/topic/nilai-kadar-normal>

Alodokter. (2024). *Hipoglikemia*. Diakses dari <https://www.alodokter.com/hipoglikemia>

LAMPIRAN

Github : <https://github.com/Mizan00-AFK/Aplikasi-BMI-dan-Kesehatan/tree/main>

Screenshot Code :

CalculationResult.kt :

```
CalculationResult.kt
1 package com.example.healthcalculator.model
2
3     6 Usages
4
5 data class CalculationResult(
6     val bmi: Double,
7     val bmiCategory: String,
8     val healthWarnings: List<String>,
9     val calorieRecommendation: String,
10    val menuRecommendation: String
11 )|
```

HealthCalculator.kt :

```
HealthCalculator.kt
1 package com.example.healthcalculator.model
2
3     2 Usages
4
5 object HealthCalculator {
6
7     1 Usage
8     fun getAgeCategory(age: Int): AgeCategory {
9         return when (age) {
10             in 0 .. 12 -> AgeCategory.CHILD
11             in 13 .. 17 -> AgeCategory.TEEN
12             in 18 .. 59 -> AgeCategory.ADULT
13             else -> AgeCategory.SENIOR
14         }
15     }
16
17     1 Usage
18     fun calculate(
19         age: Int,
20         gender: Gender,
21         heightM: Double,
22         weightKg: Double,
23         cholesterol: Double,
24         uricAcid: Double,
25         bloodSugar: Double,
26         sugarType: SugarTestType
27     ): CalculationResult {
28
29         val ageCategory = getAgeCategory(age)
30
31         // 1. Hitung BMI
32         val heightM = heightM / 100.0
33         val bmi = weightKg / (heightM * heightM)
34         val bmiCategory = when {
35             bmi < 18.5 -> "Kurus"
36             bmi < 25.0 -> "Normal"
37             bmi < 30.0 -> "Overweight"
38             else -> "Obesitas"
39         }
40
41         // 2. Evaluasi Gula Darah
42         val sugarStatus = evaluateBloodSugar(ageCategory, sugarType, value = bloodSugar)
43
44         // 3. Evaluasi Kolesterol Total
45         val cholesterolStatus = evaluateCholesterol(ageCategory, value = cholesterol)
46
47         // 4. Evaluasi Asam Urat
48         val uricStatus = evaluateUricAcid(ageCategory, gender, value = uricAcid)
49
50         // Hasil Evaluasi
51         val warnings = mutableListOf<String>()
52         if (sugarStatus != "Normal") warnings.add("Gula Darah: $sugarStatus")
53         if (cholesterolStatus != "Normal") warnings.add("Kolesterol: $cholesterolStatus")
54         if (uricStatus != "Normal") warnings.add("Asam Urat: $uricStatus")
55         if (warnings.isEmpty()) warnings.add("Semua Parameter Normal ✅")
56
57         // Rekomendasi Diet
58         val (calories, menu) = when (bmiCategory) {
59             "Kurus" -> "±2500 kkal/hari" to "Nasi, ayam, telur, susu, buah"
60             "Normal" -> "±2200 kkal/hari" to "Nasi merah, sayur, ikan, buah"
61             "Overweight" -> "±1800 kkal/hari" to "Sayur, dada ayam, oatmeal"
62             else -> "±1500 kkal/hari" to "Sayur rebus, buah, protein rendah lemak"
63         }
64
65         return CalculationResult(bmi, bmiCategory, healthWarnings = warnings, calorieRecommendation = calories, menuRecommendation = menu)
66     }
67 }
```

```
// --- LOGIKA RINCI TIAP KATEGORI ---

1 Usage
private fun evaluateBloodSugar(category: AgeCategory, type: SugarTestType, value: Double): String {
    // Batas bawah
    if (value < 60) return "Bawah: Hipoglikemia (< 60)"
    val range = when (type) {
        SugarTestType.FASTING -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN -> 70.0 .. ≤ 100.0
            AgeCategory.ADULT -> 70.0 .. ≤ 99.0
            AgeCategory.SENIOR -> 70.0 .. ≤ 110.0
        }
        SugarTestType.BEFORE_MEAL -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN, AgeCategory.ADULT -> 70.0 .. ≤ 130.0
            AgeCategory.SENIOR -> 80.0 .. ≤ 140.0
        }
        SugarTestType.AFTER_MEAL -> when (category) {
            AgeCategory.CHILD, AgeCategory.TEEN -> 0.0 .. ≤ 139.9 // < 140
            AgeCategory.ADULT, AgeCategory.SENIOR -> 0.0 .. ≤ 199.9 // < 200
        }
    }
    return when {
        value < range.start -> "Rendah (< ${range.start})"
        value > range.endInclusive -> "Tinggi (> ${range.endInclusive.toInt()})"
        else -> "Normal"
    }
}

1 Usage
private fun evaluateCholesterol(category: AgeCategory, value: Double): String {
    // Range Kadar Kolesterol
    val maxLimit = when (category) {
        AgeCategory.CHILD, AgeCategory.TEEN -> 170.0
        AgeCategory.ADULT, AgeCategory.SENIOR -> 200.0
    }
    return if (value > maxLimit) "Tinggi (> ${maxLimit.toInt()})" else "Normal"
}

1 Usage
private fun evaluateUricAcid(category: AgeCategory, gender: Gender, value: Double): String {
    val range = when (category) {
        AgeCategory.CHILD -> 2.0 .. ≤ 5.5
        AgeCategory.TEEN -> 2.5 .. ≤ 6.5
        AgeCategory.ADULT, AgeCategory.SENIOR -> if (gender == Gender.FEMALE)
            2.4 .. ≤ 6.0 else 3.4 .. ≤ 7.0
    }
    return when {
        value < range.start -> "Rendah (< ${range.start})"
        value > range.endInclusive -> "Tinggi (> ${range.endInclusive})"
        else -> "Normal"
    }
}
```

HealthEnums.kt :

The screenshot shows the Android Studio interface with the code editor open to `HealthEnums.kt`. The code defines two enums: `Gender` and `SugarTestType`, each with specific label values. The code editor highlights several usages of these enum values across the project, indicated by underlines and tool tips.

```
package com.example.healthcalculator.model

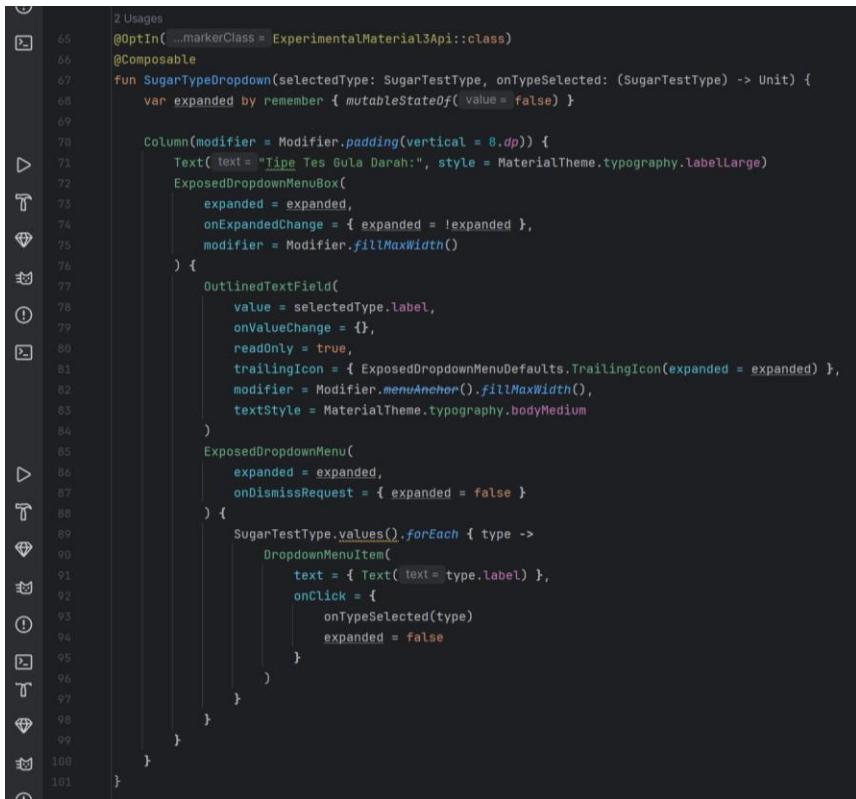
enum class Gender(val label: String) {
    MALE(label = "Laki-laki"),
    FEMALE(label = "Perempuan")
}

enum class SugarTestType(val label: String) {
    FASTING(label = "Puasa (8-9 Jam)"),
    BEFORE_MEAL(label = "Sebelum Makan"),
    AFTER_MEAL(label = "2 Jam Setelah Makan / Sewaktu")
}
```

```
① 32 Usages
14 enum class AgeCategory(val label: String) {
    6 Usages
    15     CHILD( label = "Anak-anak (0-12 th)" ),
    6 Usages
    16     TEEN( label = "Remaja (13-17 th)" ),
    6 Usages
    17     ADULT( label = "Dewasa (18-59 th)" ),
    6 Usages
    18     SENIOR( label = "Lansia (> 60 th)" )
    19 }
```

InputComponents.kt :

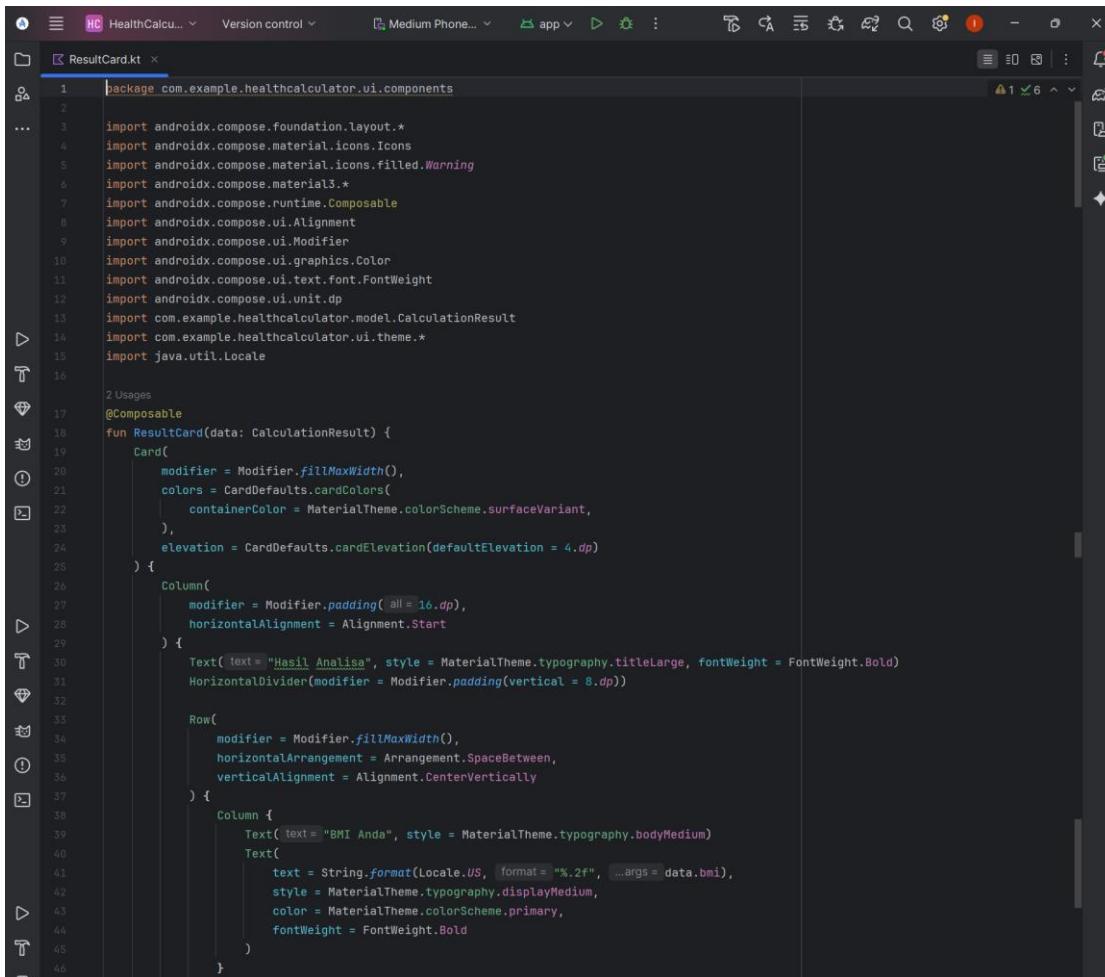
```
1 package com.example.healthcalculator.ui.components
2
3 import androidx.compose.foundation.clickable
4 import androidx.compose.foundation.layout.*
5 import androidx.compose.foundation.text.KeyboardOptions
6 import androidx.compose.material3.*
7 import androidx.compose.runtime.*
8 import androidx.compose.ui.Alignment
9 import androidx.compose.ui.Modifier
10 import androidx.compose.ui.text.input.ImeAction
11 import androidx.compose.ui.text.input.KeyboardType
12 import androidx.compose.ui.unit.dp
13 import com.example.healthcalculator.model.Gender
14 import com.example.healthcalculator.model.SugarTestType
15
16
17 7 Usages
18 @Composable
19 fun HealthInputRow(
20     label: String,
21     value: String,
22     onValueChange: (String) -> Unit,
23     imeAction: ImeAction = ImeAction.Next
24 ) {
25     OutlinedTextField(
26         value = value,
27         onValueChange = { input ->
28             if (input.all { char -> char.isDigit() || char == '.' }) {
29                 onValueChange(input)
30             }
31         },
32         label = { Text(text = label) },
33         modifier = Modifier
34             .fillMaxWidth()
35             .padding(vertical = 4.dp),
36         keyboardOptions = KeyboardOptions(
37             keyboardType = KeyboardType.Number,
38             imeAction = imeAction
39         ),
40         singleLine = true
41     )
42
43 2 Usages
44 @Composable
45 fun GenderSelection(selectedGender: Gender, onGenderSelected: (Gender) -> Unit) {
46     Column(modifier = Modifier.padding(vertical = 8.dp)) {
47         Text(text = "Jenis Kelamin:", style = MaterialTheme.typography.labelLarge)
48         Row(verticalAlignment = Alignment.CenterVertically) {
49             Gender.values().forEach { gender ->
50                 Row(
51                     verticalAlignment = Alignment.CenterVertically,
52                     modifier = Modifier
53                         .padding(end = 16.dp)
54                         .clickable { onGenderSelected(gender) }
55                 ) {
56                     RadioButton(
57                         selected = (gender == selectedGender),
58                         onClick = { onGenderSelected(gender) }
59                     )
60                     Text(text = gender.label, style = MaterialTheme.typography.bodyMedium)
61                 }
62             }
63         }
64     }
65 }
```



```
2 Usages
@OptIn( ...markerClass = ExperimentalMaterial3Api::class)
@Composable
fun SugarTypeDropdown(selectedType: SugarTestType, onTypeSelected: (SugarTestType) -> Unit) {
    var expanded by remember { mutableStateOf( value = false ) }

    Column(modifier = Modifier.padding(vertical = 8.dp)) {
        Text( text = "Tipe Tes Gula Darah:", style = MaterialTheme.typography.labelLarge)
        ExposedDropdownMenuBox(
            expanded = expanded,
            onExpandedChange = { expanded = !expanded },
            modifier = Modifier.fillMaxWidth()
        ) {
            OutlinedTextField(
                value = selectedType.label,
                onValueChange = {},
                readOnly = true,
                trailingIcon = { ExposedDropdownMenuDefaults.TrailingIcon(expanded = expanded) },
                modifier = Modifier.menuAnchor().fillMaxWidth(),
                textStyle = MaterialTheme.typography.bodyMedium
            )
            ExposedDropdownMenu(
                expanded = expanded,
                onDismissRequest = { expanded = false }
            ) {
                SugarTestType.values().forEach { type ->
                    DropdownMenuItem(
                        text = { Text( text = type.label ) },
                        onClick = {
                            onTypeSelected(type)
                            expanded = false
                        }
                    )
                }
            }
        }
    }
}
```

ResultCard.kt :

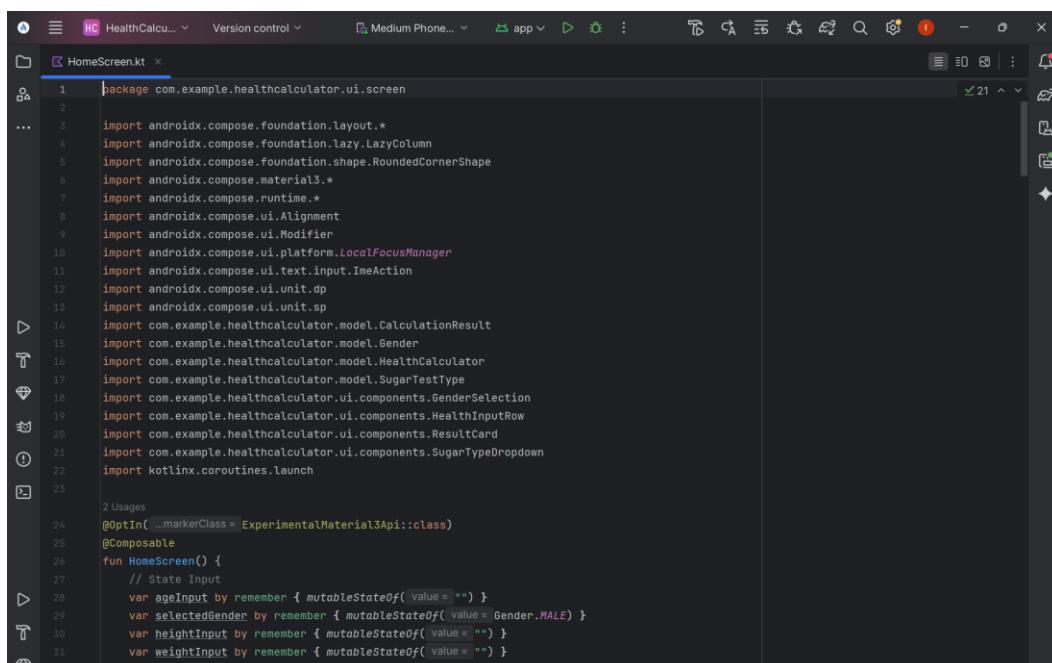


```
2 Usages
@Composable
fun ResultCard(data: CalculationResult) {
    Card(
        modifier = Modifier.fillMaxWidth(),
        colors = CardDefaults.cardColors(
            containerColor = MaterialTheme.colorScheme.surfaceVariant,
        ),
        elevation = CardDefaults.cardElevation(defaultElevation = 4.dp)
    ) {
        Column(
            modifier = Modifier.padding( all = 16.dp ),
            horizontalAlignment = Alignment.Start
        ) {
            Text( text = "Hasil Analisa", style = MaterialTheme.typography.titleLarge, fontWeight = FontWeight.Bold)
            HorizontalDivider(modifier = Modifier.padding(vertical = 8.dp))

            Row(
                modifier = Modifier.fillMaxWidth(),
                horizontalArrangement = Arrangement.SpaceBetween,
                verticalAlignment = Alignment.CenterVertically
            ) {
                Column {
                    Text( text = "BMI Anda", style = MaterialTheme.typography.bodyMedium)
                    Text(
                        text = String.format(Locale.US, format = "%.2f", ...args = data.bmi),
                        style = MaterialTheme.typography.displayMedium,
                        color = MaterialTheme.colorScheme.primary,
                        fontWeight = FontWeight.Bold
                    )
                }
            }
        }
    }
}
```

```
47     Card(
48         colors = CardDefaults.cardColors(
49             containerColor = getCategoryColor(data.bmiCategory)
50         )
51     ) {
52         Text(
53             text = data.bmiCategory,
54             modifier = Modifier.padding(horizontal = 16.dp, vertical = 8.dp),
55             color = Color.White,
56             fontWeight = FontWeight.Bold
57         )
58     }
59 }
60
61 Spacer(modifier = Modifier.height( height = 16.dp))
62
63 Text( text = "Kondisi Darah:", style = MaterialTheme.typography.labelLarge)
64 data.healthWarnings.forEach { warning ->
65     val isNormal = warning.contains( other = "Normal")
66     Row(verticalAlignment = Alignment.CenterVertically, modifier = Modifier.padding(top = 4.dp)) {
67         if (!isNormal) {
68             Icon( imageVector = Icons.Default.Warning, contentDescription = null, tint = MaterialTheme.colorScheme.error, modifier
69                 = Modifier.width( width = 4.dp))
70         }
71         Text(
72             text = warning,
73             color = if (isNormal) MaterialTheme.colorScheme.onSurface else MaterialTheme.colorScheme.error
74         )
75     }
76 }
77
78 Spacer(modifier = Modifier.height( height = 16.dp))
79
80
81 // Diet
82 Card(colors = CardDefaults.cardColors(containerColor = MaterialTheme.colorScheme.background)) {
83     Column(modifier = Modifier.padding( all = 12.dp).fillMaxWidth()) {
84         Text( text = "💡 Rekomendasi Diet", style = MaterialTheme.typography.titleMedium, color = MaterialTheme.colorScheme.primary)
85         Spacer(modifier = Modifier.height( height = 4.dp))
86         Text( text = "Target Kalori: ${data.calorieRecommendation}", fontWeight = FontWeight.SemiBold)
87         Text( text = "Menu: ${data.menuRecommendation}", style = MaterialTheme.typography.bodyMedium)
88     }
89 }
90
91 }
92
93
94 1 Usage
95 private fun getCategoryColor(category: String): Color {
96     return when (category) {
97         "Kurus" -> ColorKurus
98         "Normal" -> ColorNormal
99         "Overweight" -> ColorOverweight
100        "Obesitas" -> ColorObesitas
101        else -> Color.Gray
102    }
103 }
```

HomeScreen.kt :



The screenshot shows the Android Studio interface with the code editor open to the HomeScreen.kt file. The code is a Kotlin file for a Compose-based application. It imports various AndroidX and Compose libraries, defines a package, and contains a Composable function named HomeScreen(). The function uses remember to initialize mutableStateOf variables for ageInput, selectedGender, heightInput, and weightInput. It also imports several model classes from the com.example.healthcalculator package.

```
1 package com.example.healthcalculator.ui.screen
2
3 import androidx.compose.foundation.layout.*
4 import androidx.compose.foundation.lazy.LazyColumn
5 import androidx.compose.foundation.shape.RoundedCornerShape
6 import androidx.compose.material3.*
7 import androidx.compose.runtime.*
8 import androidx.compose.ui.Alignment
9 import androidx.compose.ui.Modifier
10 import androidx.compose.ui.platform.LocalFocusManager
11 import androidx.compose.ui.text.input.ImeAction
12 import androidx.compose.ui.unit.dp
13 import androidx.compose.ui.unit.sp
14 import com.example.healthcalculator.model.CalculationResult
15 import com.example.healthcalculator.model.Gender
16 import com.example.healthcalculator.model.HealthCalculator
17 import com.example.healthcalculator.model.SugarTestType
18 import com.example.healthcalculator.ui.components.GenderSelection
19 import com.example.healthcalculator.ui.components.HealthInputRow
20 import com.example.healthcalculator.ui.components.ResultCard
21 import com.example.healthcalculator.ui.components.SugarTypeDropdown
22 import kotlinx.coroutines.launch
23
24 2 Usages
25 @OptIn( ...markerClass = ExperimentalMaterial3Api::class)
26 @Composable
27 fun HomeScreen() {
28     // State Input
29     var ageInput by remember { mutableStateOf( value = "") }
30     var selectedGender by remember { mutableStateOf( value = Gender.MALE) }
31     var heightInput by remember { mutableStateOf( value = "") }
32     var weightInput by remember { mutableStateOf( value = "") }
33 }
```

```
var cholesterolInput by remember { mutableStateOf<String?>("0") }
var uricInput by remember { mutableStateOf<String?>("0") }

var selectedSugarType by remember { mutableStateOf<SugarTestType>(SugarTestType.FASTING) }
var sugarInput by remember { mutableStateOf<String?>("0") }

// State Output
var result by remember { mutableStateOf<CalculationResult?>(null) }

val snackbarHostState = remember { SnackbarHostState() }
val scope = rememberCoroutineScope()
val focusManager = LocalFocusManager.current

Scaffold {
    scaffoldBar {
        snackbarHost = { SnackbarHost(snackbarHostState) },
        topBar = {
            CenterAlignedTopAppBar(
                title = { Text(text = "Health & Diet Planner") },
                colors = TopAppBarDefaults.centerAlignedTopAppBarColors(
                    containerColor = MaterialTheme.colorScheme.primaryContainer,
                    titleContentColor = MaterialTheme.colorScheme.onPrimaryContainer
                )
            )
        }
    } { paddingValues ->
        LazyColumn(
            modifier = Modifier.padding(paddingValues).padding(all = 16.dp).fillMaxSize(),
            horizontalAlignment = Alignment.CenterHorizontally
        ) {
            item {
                Text(
                    text = "Data Pribadi",
                    style = MaterialTheme.typography.titleMedium,
                    color = MaterialTheme.colorScheme.primary,
                    modifier = Modifier.fillMaxWidth().padding(bottom = 8.dp)
                )
            }

            // --- Input Data Diri ---
            item {
                HealthInputRow(label = "Umur (Tahun)", value = ageInput, onValueChange = { ageInput = it })
                GenderSelection(selectedGender = it)
                Row(modifier = Modifier.fillMaxWidth(), horizontalArrangement = Arrangement.spacedBy(space = 8.dp)) {
                    Box(modifier = Modifier.weight(weight = 1f)) {
                        HealthInputRow(label = "Tinggi (cm)", value = heightInput, onValueChange = { heightInput = it })
                    }
                    Box(modifier = Modifier.weight(weight = 1f)) {
                        HealthInputRow(label = "Berat (kg)", value = weightInput, onValueChange = { weightInput = it })
                    }
                }
            }

            item {
                Spacer(modifier = Modifier.height(16.dp))
                Text(
                    text = "Hasil Lab",
                    style = MaterialTheme.typography.titleMedium,
                    color = MaterialTheme.colorScheme.primary,
                    modifier = Modifier.fillMaxWidth().padding(bottom = 8.dp)
                )
            }

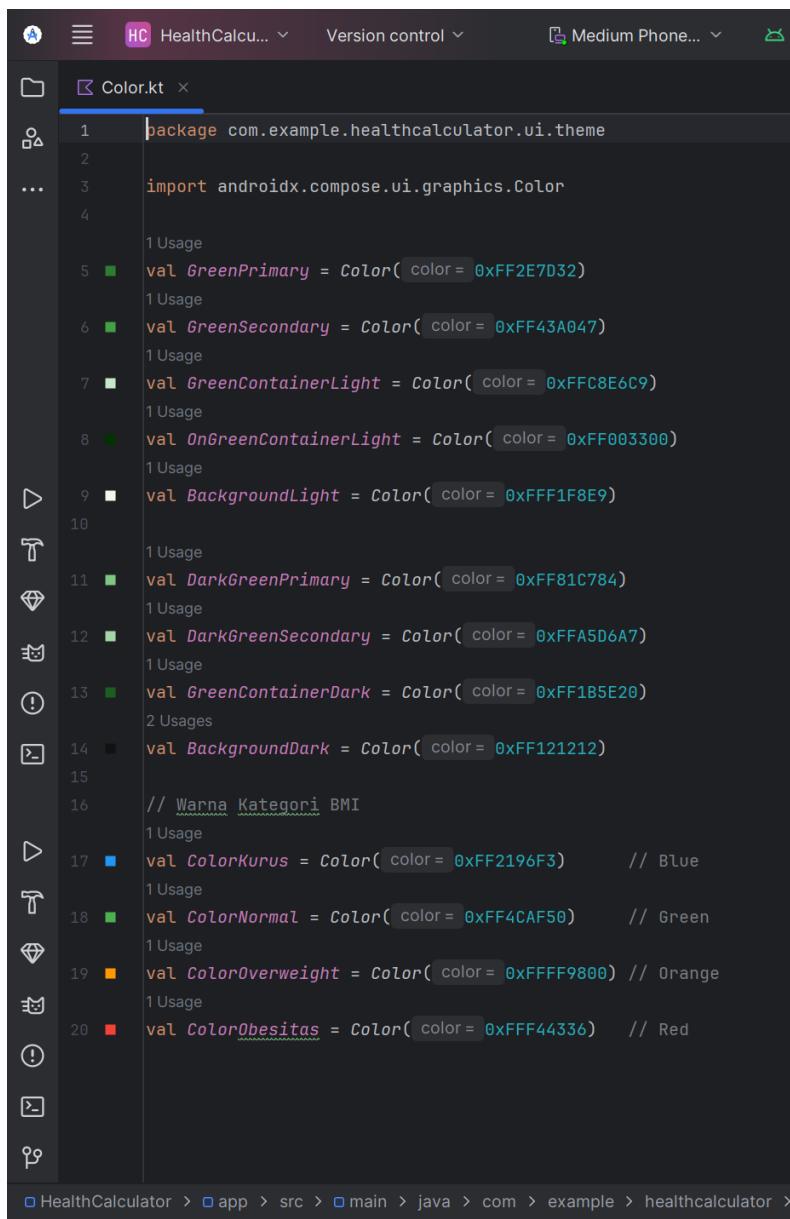
            // --- Input Data Lab/Kesehatan ---
            item {
                SugarTypeDropdown(selectedType = selectedSugarType)
                HealthInputRow(label = "Gula Darah (mg/dL)", value = sugarInput, onValueChange = { sugarInput = it })
                HealthInputRow(label = "Kolesterol Total (mg/dL)", value = cholesterolInput, onValueChange = { cholesterolInput = it })
                HealthInputRow(label = "Asam Urat (mg/dL)", value = uricInput, onValueChange = { uricInput = it }, ImeAction.Done)
            }

            // --- Tombol Hitung ---
            item {
                Spacer(modifier = Modifier.height(24.dp))
                Button(
                    onClick = {
                        focusManager.clearFocus()
                        // Validasi
                        val age = ageInput.toIntOrNull()
                        val h = heightInput.toDoubleOrNull()
                        val w = weightInput.toDoubleOrNull()
                        val chol = cholesterolInput.toDoubleOrNull()
                        val uric = uricInput.toDoubleOrNull()
                        val sugar = sugarInput.toDoubleOrNull()

                        if (age != null && h != null && w != null && chol != null && uric != null && sugar != null) {
                            result = HealthCalculator.calculate(
                                age, selectedGender, heightCm = h, weightKg = w, cholesterol = chol, uricAcid = uric, bloodSugar = sugar, selectedSugarType
                            )
                        } else {
                            scope.launch { snackbarHostState.showSnackbar(message = "Mohon lengkapi semua data dengan benar.") }
                        }
                    },
                    modifier = Modifier.fillMaxWidth().height(height = 50.dp),
                    shape = RoundedCornerShape(size = 12.dp)
                )
            }

            // --- Hasil ---
            item {
                result?.let { res ->
                    Spacer(modifier = Modifier.height(24.dp))
                    ResultCard(data = res)
                }
            }
        }
    }
}
```

Color.kt :



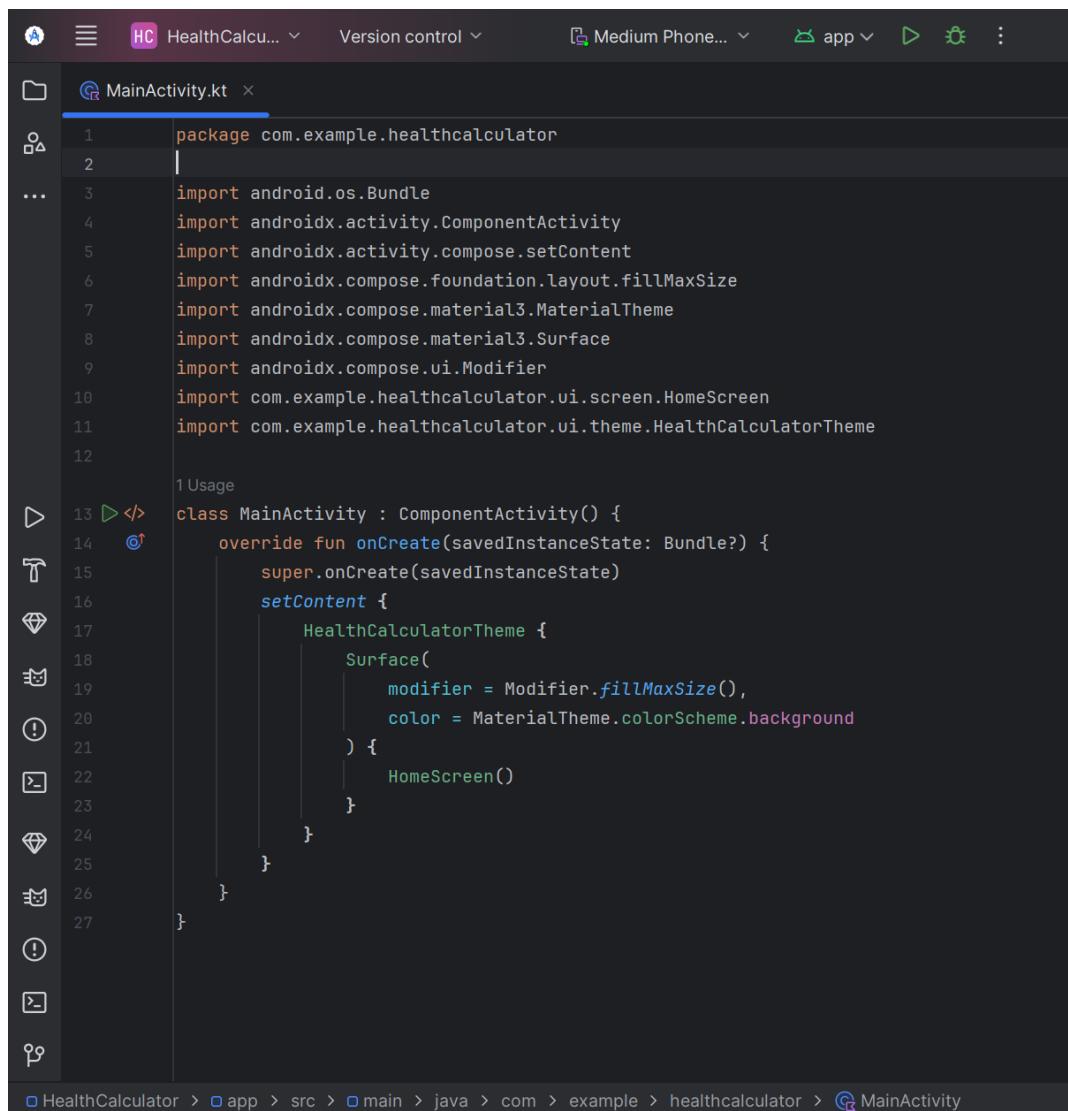
```
1 package com.example.healthcalculator.ui.theme
2
3 import androidx.compose.ui.graphics.Color
4
5 val GreenPrimary = Color( color = 0xFF2E7D32)
6 val GreenSecondary = Color( color = 0xFF43A047)
7 val GreenContainerLight = Color( color = 0xFFC8E6C9)
8 val OnGreenContainerLight = Color( color = 0xFF003300)
9 val BackgroundLight = Color( color = 0xFFFF1F8E9)
10
11 val DarkGreenPrimary = Color( color = 0xFF81C784)
12 val DarkGreenSecondary = Color( color = 0xFFA5D6A7)
13 val GreenContainerDark = Color( color = 0xFF1B5E20)
14 val BackgroundDark = Color( color = 0xFF121212)
15
16 // Warna Kategori BMI
17 val ColorKurus = Color( color = 0xFF2196F3)      // Blue
18 val ColorNormal = Color( color = 0xFF4CAF50)      // Green
19 val ColorOverweight = Color( color = 0xFFFF9800)   // Orange
20 val ColorObesitas = Color( color = 0xFFFF4433)    // Red
```

Theme.kt :

The screenshot shows the Android Studio interface with the code editor open to the `Theme.kt` file. The file contains Kotlin code for defining a theme in a Material 3 style. It includes definitions for `DarkColorScheme` and `LightColorScheme`, and a `HealthCalculatorTheme` function that returns a `MaterialTheme`. The code uses `Material3` components and `WindowCompat` for styling.

```
1 package com.example.healthcalculator.ui.theme
2
3 import android.app.Activity
4 import androidx.compose.foundation.isSystemInDarkTheme
5 import androidx.compose.material3.*
6 import androidx.compose.runtime.Composable
7 import androidx.compose.runtime.SideEffect
8 import androidx.compose.ui.graphics.Color
9 import androidx.compose.ui.graphics.toArgb
10 import androidx.compose.ui.platform.LocalView
11 import androidx.core.view.WindowCompat
12
13 1 Usage
14 private val DarkColorScheme = darkColorScheme(
15     primary = DarkGreenPrimary,
16     onPrimary = Color.Black,
17     primaryContainer = GreenContainerDark,
18     onPrimaryContainer = Color.White,
19     secondary = DarkGreenSecondary,
20     surface = BackgroundDark,
21     background = BackgroundDark
22
23 1 Usage
24 private val LightColorScheme = lightColorScheme(
25     primary = GreenPrimary,
26     onPrimary = Color.White,
27     primaryContainer = GreenContainerLight,
28     onPrimaryContainer = OnGreenContainerLight,
29     secondary = GreenSecondary,
30     surface = BackgroundLight,
31     background = Color.White
32
33 2 Usages
34 @Composable
35 fun HealthCalculatorTheme(
36     darkTheme: Boolean = isSystemInDarkTheme(),
37     content: @Composable () -> Unit
38 ) {
39     val colorScheme = if (darkTheme) DarkColorScheme else LightColorScheme
40     val view = LocalView.current
41
42     if (!view.isInEditMode) {
43         SideEffect {
44             val window = (view.context as Activity).window
45             window.statusBarColor = colorScheme.primary.toArgb()
46             WindowCompat.getInsetsController(window, view).isAppearanceLightStatusBars = !darkTheme
47         }
48     }
49
50     MaterialTheme(
51         colorScheme = colorScheme,
52         content = content
53     )
54 }
```

MainActivity.kt :



The screenshot shows the Android Studio code editor for the file `MainActivity.kt`. The code is written in Kotlin and defines a main activity that uses the Compose library to set up a home screen.

```
1 package com.example.healthcalculator
2
3 import android.os.Bundle
4 import androidx.activity.ComponentActivity
5 import androidx.activity.compose.setContent
6 import androidx.compose.foundation.layout.fillMaxSize
7 import androidx.compose.material3.MaterialTheme
8 import androidx.compose.material3.Surface
9 import androidx.compose.ui.Modifier
10 import com.example.healthcalculator.ui.screen.HomeScreen
11 import com.example.healthcalculator.ui.theme.HealthCalculatorTheme
12
13 class MainActivity : ComponentActivity() {
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContent {
17             HealthCalculatorTheme {
18                 Surface(
19                     modifier = Modifier.fillMaxSize(),
20                     color = MaterialTheme.colorScheme.background
21                 ) {
22                     HomeScreen()
23                 }
24             }
25         }
26     }
27 }
```

The code editor includes standard Android Studio features like a toolbar at the top, a navigation bar on the left, and a status bar at the bottom. The status bar shows the project path: `HealthCalculator > app > src > main > java > com > example > healthcalculator > MainActivity`.