# PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN




Disusun Oleh:

N a m a : Muhammad Harel

NIM : 21523032

**PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA**

**FAKULTAS TEKNOLOGI INDUSTRI**

**UNIVERSITAS ISLAM INDONESIA**

**2025**

**HALAMAN PENGESAHAN DOSEN PEMBIMBING**

# PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN

## TUGAS AKHIR

Disusun Oleh:


N a m a : Muhammad Harel

NIM : 21523032


Yogyakarta, 1 Mei 2025

Pembimbing,

( Sri Mulyati,S.Kom., M.Kom. )

**HALAMAN PENGESAHAN DOSEN PENGUJI**

# PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN

## TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 23 Mei 2025

Tim Penguji

Sri Mulyati, S.Kom., M.Kom.

**Anggota 1**

Hari Setiaji, S.Kom., M.Eng.

**Anggota 2**

Dr. Nur Wijayaning Rahayu, S.Kom., M.Cs.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

( Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. )

**HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR**

Yang bertanda tangan di bawah ini:

Nama : Muhammad Harel

NIM : 21523032

Tugas akhir dengan judul:

# PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apa pun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 8 Mei 2025




( Muhammad Harel )

## HALAMAN PERSEMBAHAN

Segala puji dan syukur saya panjatkan kepada Allah SWT yang telah memberikan segala nikmat dan karunia-Nya, sehingga saya dapat menyelesaikan tugas akhir yang berjudul "PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN ". Saya ingin mengucapkan terima kasih yang sebesar-besarnya kepada orang tua, keluarga, dan teman-teman dekat yang selalu memberikan dukungan dan motivasi selama proses penulisan tugas akhir ini. Ucapan terima kasih juga saya sampaikan kepada Ibu Sri Mulyati, S.Kom., M.Kom., selaku dosen pembimbing saya yang dengan sabar memberikan arahan, bimbingan, dan semangat dalam menjalani penelitian ini.

## HALAMAN MOTO

“Allah tidak membebani seseorang melainkan sesuai dengan kemampuannya”

(QS Al-Baqarah: 286)

"*Success is* 1% *inspiration,* 99% *perspiration*."

(Thomas Edison)

# KATA PENGANTAR

Segala puji dan syukur hanya milik Allah SWT, yang senantiasa memberikan rahmat, berkah, dan nikmat kepada seluruh hamba-Nya. Semoga shalawat dan salam selalu tercurahkan kepada Nabi Muhammad SAW, suri tauladan yang menjadi petunjuk hidup umat manusia.

Dengan izin dan petunjuk dari Allah SWT, saya berhasil menyelesaikan tugas akhir yang berjudul " PENGEMBANGAN APLIKASI ANDROID DENGAN INTEGRASI AI UNTUK MONITORING KONSUMSI GULA HARIAN ". Tugas akhir ini merupakan salah satu syarat dalam penyelesaian studi Sarjana pada Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia.

Selama proses penyelesaian tugas akhir ini, saya ingin menyampaikan rasa terima kasih yang mendalam kepada semua pihak yang telah memberikan dukungan dan bantuan. Dengan tulus saya mengucapkan terima kasih kepada:

1. Allah SWT, yang telah memberikan petunjuk dan memudahkan penulis dalam menyelesaikan laporan ini.
2. Kedua Orang Tua, Aswandi dan Feri Trisna, yang telah memberikan kasih sayang tanpa henti, dukungan, dan doa terbaik sepanjang perjalanan penulis. Terima kasih atas segala cinta, pengorbanan, dan dorongan yang selalu menyertai setiap langkah penulis. Kalian adalah inspirasi dan sumber kekuatan utama bagi penulis dalam mencapai setiap pencapaian ini.
3. Bapak Dr. Raden Teduh Dirgahayu, S.T., M.Sc., sebagai Kepala Jurusan Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
4. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Universitas Islam Indonesia.
5. Ibu Sri Mulyati, S.Kom., M.Kom., sebagai dosen pembimbing yang dengan sabar memberikan arahan, bimbingan, dan motivasi sepanjang penelitian ini.
6. Seluruh dosen Program Studi Informatika yang telah memberikan ilmu kepada saya selama proses penyusunan tugas akhir.
7. Teman-teman yang telah meluangkan waktu untuk menemani dan memberikan semangat dalam proses pengerjaan tugas akhir ini.
8. Semua pihak yang telah mendukung dalam pelaksanaan tugas akhir yang tidak bisa saya sebutkan satu per satu.

Saya menyadari bahwa tugas akhir ini masih jauh dari sempurna. Oleh karena itu, saya sangat mengharapkan adanya saran dan kritik yang konstruktif agar karya ini bisa menjadi

lebih baik di masa yang akan datang. Semoga aplikasi yang telah dikembangkan dapat memberikan manfaat bagi masyarakat, khususnya dalam membantu memantau konsumsi gula harian guna mencegah diabetes. Harapan saya, penelitian ini dapat berkontribusi positif bagi perkembangan ilmu pengetahuan dan teknologi di masa depan.

Yogyakarta, 8 Mei 2025



( Muhammad Harel)

# SARI

Diabetes melitus telah menjadi masalah kesehatan yang signifikan di seluruh dunia, termasuk Indonesia, dengan jumlah penderita yang terus meningkat setiap tahunnya. Berdasarkan data dari International Diabetes Federation (IDF), pada tahun 2021, jumlah penderita diabetes global mencapai 537 juta jiwa dan diprediksi akan meningkat menjadi 783 juta jiwa pada tahun 2045, dengan Indonesia menempati posisi kelima dengan 19,5 juta jiwa penderita diabetes pada tahun 2021. Salah satu faktor utama yang mempengaruhi peningkatan jumlah penderita diabetes adalah pola konsumsi gula yang berlebihan, yang dipicu oleh gaya hidup tidak sehat, seperti konsumsi makanan tinggi gula dan lemak serta rendahnya tingkat aktivitas fisik. Hal ini semakin terlihat di wilayah perkotaan, di mana konsumsi makanan olahan dan minuman manis cenderung lebih tinggi. Untuk itu, pengendalian konsumsi gula menjadi salah satu langkah pencegahan yang penting dalam mengurangi resiko diabetes. Namun, meskipun regulasi mengenai informasi kandungan gula pada produk pangan telah diterapkan, tingkat pemahaman masyarakat terhadap informasi tersebut masih sangat rendah. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan aplikasi berbasis Android yang dapat membantu masyarakat memantau konsumsi gula harian mereka dengan lebih efektif, serta memberikan informasi edukatif mengenai batas konsumsi gula yang aman sesuai dengan rekomendasi World Health Organization (WHO) dan Kementerian Kesehatan Republik Indonesia. Aplikasi ini tidak hanya berfungsi sebagai alat pemantauan, tetapi juga memberikan saran dan notifikasi kepada pengguna untuk membantu mereka mengelola asupan gula dengan lebih baik, sehingga diharapkan dapat mengurangi resiko diabetes dalam jangka panjang. Metodologi yang digunakan dalam penelitian ini adalah pendekatan Agile, yang memungkinkan pengembangan dilakukan secara iteratif dan responsif terhadap kebutuhan pengguna. Pengujian aplikasi dilakukan dengan menggunakan Black Box Testing, yang berfokus pada pengujian fungsionalitas aplikasi tanpa melihat struktur internalnya. Hasil pengujian ini memastikan bahwa aplikasi berjalan sesuai dengan skenario yang diinginkan. Selain itu, aplikasi juga diuji menggunakan System Usability Scale (SUS), yang menunjukkan aplikasi memperoleh skor rata-rata 85, yang termasuk dalam kategori grade "B" dengan penilaian "Excellent" pada aspek kegunaan. Dengan demikian, aplikasi ini telah terbukti efektif dalam memenuhi tujuan utama untuk memantau konsumsi gula harian dan meningkatkan kesadaran kesehatan pengguna.

Kata Kunci: Aplikasi Android, Pemantauan Konsumsi Gula, Diabetes, Metodologi Agile, System Usability Scale, Pengujian Aplikasi.

# GLOSARIUM

| | |
|---|---|
| Agile Development | Metodologi pengembangan perangkat lunak yang fleksibel dan iteratif, memungkinkan perubahan cepat berdasarkan umpan balik pengguna secara langsung. |
| Android | Sistem operasi untuk perangkat mobile, dikembangkan oleh Google, yang digunakan untuk membuat aplikasi mobile. |
| Express.js | Framework berbasis Node.js untuk membangun aplikasi web dan API, memudahkan pengelolaan routing dan permintaan HTTP. |
| Black Box Testing | Pengujian perangkat lunak yang berfokus pada fungsionalitas aplikasi tanpa melihat struktur internalnya. |
| Diabetes Mellitus | Penyakit yang ditandai oleh tingginya kadar gula darah, sering disebabkan oleh gangguan insulin atau gaya hidup tidak sehat. |
| Google Cloud | Platform cloud dari Google yang menyediakan berbagai layanan untuk penyimpanan data, komputasi, dan machine learning. |
| Retrofit | Pustaka di Android untuk memudahkan komunikasi dengan server melalui HTTP, sering digunakan untuk mengakses data dari API. |

# DAFTAR ISI

HALAMAN JUDUL ..... i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ..... ii
HALAMAN PENGESAHAN DOSEN PENGUJI ..... iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR ..... iv
HALAMAN PERSEMBAHAN ..... v
HALAMAN MOTO ..... vi
KATA PENGANTAR ..... vii
SARI ..... ix
GLOSARIUM ..... x
DAFTAR ISI ..... xi
DAFTAR TABEL ..... xiii
DAFTAR GAMBAR ..... xiv
BAB I PENDAHULUAN ..... 1
1.1 Latar belakang ..... 1
1.2 Rumusan Masalah ..... 2
1.3 Batas Masalah ..... 2
1.4 Tujuan Penelitian ..... 3
1.5 Manfaat Penelitian ..... 3
1.6 Sistematika Penulisan ..... 4
BAB II LANDASAN TEORI ..... 5
2.1 Diabetes Mellitus ..... 5
2.2 Pengendalian Konsumsi Gula ..... 5
2.3 Analisis Konsumsi Gula Berbasis Kalori ..... 6
2.4 Teknologi Pengembangan Aplikasi ..... 7
2.5 Agile Development ..... 8
2.6 Perbandingan Metode Pengembangan ..... 10
2.7 Tinjauan Pustaka Fitur Aplikasi ..... 16
BAB III METODOLOGI ..... 19
3.1 Model Deteksi AI ..... 20
3.2 Pegembangan Aplikasi ..... 21
BAB IV HASIL DAN PEMBAHSAN ..... 30
4.1 Tahap Survei ..... 30
4.2 Analisis Kebutuhan ..... 33

4.3 Tahap perancangan (plan) ..... 35
4.4 Tahap Rancangan (*Design*) ..... 37
4.5 Pegembangan ..... 62
4.6 Hasil Pegujian ..... 96
BAB V KESIMPULAN DAN SARAN ..... 102
5.1 Kesimpulan ..... 102
5.2 Saran ..... 102
DAFTAR PUSTAKA ..... 104

# DAFTAR TABEL

Tabel 3. 1 Daftar Pertanyaan Kepada Calon Pengguna ..... 22
Tabel 3. 2 Pertanyaan Kuesioner Pengujian System Usability Scale (SUS) ..... 28
Tabel 3. 3 Tabel Skor Penilaian Skala Likert ..... 28
Tabel 4.1 Hasil Survai ..... 30
Tabel 4. 2 Sprint ..... 35
Tabel 4. 3 Aktor pada *Use Case Diagram* ..... 37
Tabel 4. 4 Fungsionalitas Fitur ..... 38
Tabel 4. 5 Implementasi API ..... 95
Tabel 4. 6 Hasil Black Box Testing ..... 97
Tabel 4. 7 Tabel Hasil Kuesioner Pengujian SUS ..... 99
Tabel 4. 8 Tabel Hasil Perhitungan SUS ..... 100
Tabel 4. 9 Tabel Hasil lima Variabel Usability ..... 100

# DAFTAR GAMBAR

Gambar 3. 1 Proses pembuatan aplikasi dan implementasi model ..... 19
Gambar 3. 2 Metodelogi Agile ..... 21
Gambar 3. 3 *Cloud Architecture* ..... 24
Gambar 3. 4 Arsitektur Sistem ..... 25
Gambar 3. 5 Arsitektur Perangkat Lunak ..... 27
Gambar 3. 6 Skala Skor Pengujian SUS ..... 29
Gambar 4. 1 Gantt Chart ..... 35
Gambar 4. 2 *Use Case Diagram* ..... 37
Gambar 4. 3 *Activity Diagram* Login ..... 40
Gambar 4. 4 *Activity Diagram* Informasi Pengguna ..... 41
Gambar 4. 5 *Activity Diagram* Data Gula Otomatis ..... 42
Gambar 4. 6 *Activity Diagram* Gula Manual ..... 43
Gambar 4. 7 *Activity Diagram* Melihat Riwayat Konsumsi ..... 44
Gambar 4. 8 *Activity Diagram* Melihat Statistik ..... 45
Gambar 4. 9 *Activity Diagram* Menggunakan Chat Bot ..... 46
Gambar 4. 10 *Activity Diagram* Melihat Edukasi ..... 46
Gambar 4. 11 *Activity Diagram* Melihat Hasil Informasi Penguna ..... 47
Gambar 4. 12 *Activity Diagram* Notifikasi ..... 48
Gambar 4. 13 *Activity Diagram* Mengubah Email ..... 49
Gambar 4. 14 *Activity Diagram* Mengubah Username ..... 50
Gambar 4. 15 *Activity Diagram* Mengubah Password ..... 51
Gambar 4. 16 *Activity Diagram* Scan Bar*Code* ..... 52
Gambar 4. 17 Rancangan Database ..... 53
Gambar 4. 18 *Wireframe Splash Screen* dan Onbording ..... 54
Gambar 4. 19 *Wireframe* Login ..... 54
Gambar 4. 20 *Wireframe* Register ..... 54
Gambar 4. 21 *Wireframe* Informasi Penguna ..... 55
Gambar 4. 22 *Wireframe* Home ..... 56
Gambar 4. 23 *Wireframe* Detail Informasi ..... 56
Gambar 4. 24 *Wireframe* Detail Artikel ..... 57
Gambar 4. 25 *Wireframe* Pencatatan Manual ..... 57
Gambar 4. 26 *Wireframe* Pencataan Otomatis ..... 58
Gambar 4. 27 *Wireframe* Hasil Pencatatan Otomatis ..... 58

Gambar 4. 28 *Wireframe* Chat Bot ..... 59
Gambar 4. 29 *Wireframe* Laporan ..... 59
Gambar 4. 30 *Wireframe* Statistik ..... 60
Gambar 4. 31 *Wireframe* Profile ..... 60
Gambar 4. 32 *Wireframe* Account ..... 61
Gambar 4. 33 *Wireframe* Password ..... 61
Gambar 4. 34 *Wireframe* hasil Informasi Pengguna ..... 62
Gambar 4. 35 Configurasi Api ..... 63
Gambar 4. 36 *Splash Screen* ..... 64
Gambar 4. 37 Onbording ..... 64
Gambar 4. 38 Login ..... 65
Gambar 4. 39 *Code* Halaman Login ..... 66
Gambar 4. 40 Register ..... 66
Gambar 4. 41 *Code* Halaman Register ..... 67
Gambar 4. 42 Informasi Penguna ..... 68
Gambar 4. 43 *Code* Halaman Informasi Penguna ..... 70
Gambar 4. 44 Home ..... 71
Gambar 4. 45 *Code* Halaman Home ..... 74
Gambar 4. 46 Detail Informasi ..... 75
Gambar 4. 47 Edukasi ..... 76
Gambar 4. 48 Detail Edukasi ..... 77
Gambar 4. 49 Pencatatan Manual ..... 77
Gambar 4. 50 *Code* Halaman Pencatatan Manual ..... 79
Gambar 4. 51 Pencatatan Otomatis ..... 80
Gambar 4. 52 *Code* Pencatatan Otomatis ..... 82
Gambar 4. 53 Hasil Pencatatan Otomatis ..... 82
Gambar 4. 54 *Code* Hasil Pencatatan Otomatis ..... 84
Gambar 4. 55 Chat Bot ..... 84
Gambar 4. 56 *Code* Halaman Chat Bot ..... 85
Gambar 4. 57 Statistik ..... 86
Gambar 4. 58 *Code* Halaman Statistik ..... 87
Gambar 4. 59 Laporan Riwayat Harian ..... 88
Gambar 4. 60 *Code* Halaman Laporan Riwayat Harian ..... 90
Gambar 4. 61 Profile ..... 90

Gambar 4. 62 Account ..... 91
Gambar 4. 63 Password ..... 91
Gambar 4. 64 Hasil Informasi Penguna ..... 92
Gambar 4. 65 Bar*Code* Scan ..... 93
Gambar 4. 66 *Code* Halaman Bar*Code* Scan ..... 95

# BAB I
# PENDAHULUAN

## 1.1 Latar belakang

Kesehatan merupakan faktor penting dalam kehidupan manusia, terutama di era globalisasi. Pada sektor kesehatan, kualitas hidup suatu negara sangat bergantung pada kemampuan masyarakat dalam menjaga kesehatannya, terutama melalui pencegahan dan pengelolaan penyakit kronis. Diabetes mellitus, yang ditandai dengan hiperglikemia akibat gangguan produksi insulin oleh pankreas atau rendahnya sensitivitas sel terhadap insulin, menjadi salah satu masalah kesehatan global (IDF, 2021)

Diabetes mellitus telah menjadi salah satu masalah kesehatan terbesar di Indonesia dan dunia. Berdasarkan data yang diberikan oleh *Internation Doeabetes Federatin* (IDF) jumlah penderta diabetes di dunia setiap tahun nya mengelami peningkatan bahkan diproyeksikan hingga tahun 2045 terus meningkat. Per tahun 2021 jumalah penderita diabetes sudah mencapai 537 juta jiwa. Angka ini diprediksi akan terus meningkat mencapai 643 juta di tahun 2030 dan 783 juta pada tahun 2045. Indonesia menduduki peringkat kelima negara dengan jumlah diabetes terbanyak dengan 19,5 juta penderita di tahun 2021 dan diprediksi akan menjadi 28,6 juta pada 2045 (IDF, 2021).

Kondisi ini dipengaruhi oleh berbagai faktor, termasuk genetika, gaya hidup yang tidak aktif (sedentari), obesitas, serta konsumsi makanan dengan kandungan gula dan lemak yang tinggi (Kementerian Kesehatan, 2020). Berdasarkan data Riskesdas 2018, prevalensi diabetes di Indonesia mencapai 2%, dengan angka yang lebih tinggi di perkotaan dibandingkan dengan pedesaan. Hal ini mencerminkan perubahan gaya hidup dan pola makan masyarakat, terutama di perkotaan yang lebih cenderung mengkonsumsi makanan olahan tinggi gula dan rendah serat.

Diabetes melitus, terutama diabetes tipe-2, sering kali disebabkan oleh kebiasaan gaya hidup yang tidak sehat, salah satunya adalah konsumsi gula yang berlebihan. Konsumsi gula yang tinggi dapat meningkatkan resistensi insulin dan menyebabkan berbagai komplikasi kesehatan lainnya, seperti penyakit jantung, obesitas, dan hipertensi. Menurut data World Health Organization (WHO), konsumsi gula yang berlebihan merupakan salah satu kontributor utama meningkatnya prevalensi penyakit tidak menular di dunia, termasuk di Indonesia (World Health Organization, 2023)

Dalam upaya mengatasi masalah ini, pemerintah Indonesia melalui Kementerian Kesehatan telah mengeluarkan Peraturan Menteri Kesehatan Nomor 30 Tahun 2013 mengenai

Pencantuman Informasi Kandungan Gula, Garam, dan Lemak serta Pesan Kesehatan pada Pangan Olahan dan Pangan Siap Saji. Peraturan tersebut bertujuan agar masyarakat dapat mengetahui informasi nilai gizi yang terkandung di makanan dan minuman yang dituliskan dalam label makanan(Kementerian Kesehatan Republik Indonesia, 2013). Namun, meskipun regulasi ini telah diterapkan, tingkat pemahaman masyarakat terhadap informasi gizi masih rendah. Penelitian yang dilakukan oleh Mauludyani menunjukkan bahwa hanya 37,5% konsumen yang benar-benar memperhatikan informasi gizi sebelum membeli makanan (Mauludyani et al., 2021).

Kesenjangan antara ketersediaan informasi gizi dan pemahaman masyarakat terhadap informasi tersebut menunjukkan perlunya solusi yang lebih mudah diakses dan digunakan oleh masyarakat. Salah satu pendekatan yang dapat digunakan adalah dengan memanfaatkan teknologi digital. Penggunaan aplikasi berbasis Android untuk memantau konsumsi gula harian merupakan solusi yang relevan dengan meningkatnya penetrasi smartphone di Indonesia sesuai dengan tren meningkatnya penetrasi smartphone di Indonesia, yang mana jumlah pengguna aktif mencapai 209,3 juta jiwa pada tahun 2023 (Andalas, 2024). Aplikasi ini diharapkan dapat memberikan informasi mengenai kandungan gula dalam makanan dan membantu pengguna mengontrol konsumsi gula mereka, yang pada akhirnya dapat membantu mengurangi resiko diabetes.

Selain itu, aplikasi ini tidak hanya berfungsi sebagai alat pemantauan, tetapi juga sebagai sarana edukasi bagi masyarakat. Dengan memudahkan akses informasi mengenai kadar gula pada berbagai jenis makanan, diharapkan aplikasi ini dapat meningkatkan kesadaran masyarakat akan pentingnya pengendalian asupan gula. Aplikasi semacam ini sangat penting mengingat tren konsumsi makanan dan minuman manis yang meningkat, khususnya di kalangan generasi muda.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut , maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana mengembangkan aplikasi Android yang dapat mengintegrasikan sistem deteksi makanan berbasis kecerdasan buatan (AI)?
2. Bagaimana mengembangkan aplikasi Android untuk pencatatan dan analisis konsumsi gula harian dengan visualisasi grafik interaktif yang dapat memberikan umpan balik edukatif sesuai dengan indikator batas anjuran konsumsi

## 1.3 Batas Masalah

Penelitian ini dibatasi pada:

1. Penelitian ini hanya mencakup remaja dan dewasa muda, yaitu individu berusia 13–35 tahun, sebagai target pengguna aplikasi.
2. Pengembangan aplikasi berbasis Android ini difokuskan untuk mencatat dan memantau konsumsi gula harian pada makanan dan minuman yang umumnya dikonsumsi oleh kelompok usia tersebut.
3. Data makanan dan kandungan gula yang digunakan dalam aplikasi terbatas pada makanan yang umum dikonsumsi oleh remaja dan dewasa muda tersebut.
4. Penelitian ini tidak mencakup pengujian klinis terhadap kadar gula darah pengguna, melainkan berfokus pada edukasi dan pencatatan asupan gula berdasarkan rekomendasi batas konsumsi gula harian menurut WHO dan Kementerian Kesehatan RI.

## 1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Mengembangkan sebuah sistem berbasis Android yang dapat menyediakan informasi kadar gula pada makanan.
2. Mengimplementasikan sistem yang memungkinkan pengguna untuk memantau konsumsi gula harian mereka.
3. Mengukur efektivitas sistem dalam membantu pengguna mengontrol asupan gula harian dan mengurangi resiko diabetes.
4. Memungkinkan pengguna untuk mencatat asupan gula dan mendapatkan notifikasi serta analisis konsumsi.
5. Mengintegrasikan aplikasi dengan model deteksi makanan berbasis gambar yang dikembangkan oleh rekan peneliti.

## 1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat bagi berbagai pihak, yaitu:

1. **Bagi Pengguna**

   Memudahkan pemantauan konsumsi gula harian melalui aplikasi berbasis Android yang menyediakan informasi kadar gula dalam makanan serta pengingat untuk mengontrol asupan gula.

2. **Bagi Pengembang**

   Menjadi referensi dalam pengembangan aplikasi kesehatan berbasis mobile, khususnya dalam penerapan **metode Agile** dan integrasi fitur **notifikasi, pelacakan konsumsi, serta analisis data**.

3. **Bagi Akademisi**

   Memberikan dasar bagi penelitian lanjutan terkait pemantauan pola makan berbasis teknologi dan pengembangan intervensi digital dalam kesehatan.

## 1.6 Sistematika Penulisan

Struktur penulisan yang digunakan dalam penyusunan tugas akhir ini dapat dijelaskan sebagai berikut:

**BAB I PENDAHULUAN**

Bab ini menguraikan latar belakang penelitian, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan tugas akhir secara rinci.

**BAB II LANDASAN TEORI**

Bab ini memuat teori-teori dan konsep-konsep yang relevan dengan penelitian, serta mendukung analisis pemecahan masalah yang diajukan. Selain itu, kajian pustaka juga mengulas penelitian terdahulu yang berkaitan dengan topik penelitian ini.

**BAB III METODELOGI**

Bab ini menjelaskan secara rinci metode penelitian yang digunakan, termasuk penerapan teori dalam pengembangan aplikasi yang dikembangkan serta langkah-langkah penelitian yang dilakukan.

**BAB IV HASIL DAN PEMBAHASAN**

Bab ini memaparkan hasil penelitian yang telah dilakukan, dengan penyajian dalam bentuk diagram alur, potongan kode program, tabel data, gambar, atau format lain yang sesuai. Setiap hasil akan dibahas secara sistematis berdasarkan tahapan penelitian.

**BAB V KESIMPULAN DAN SARAN**

Bab ini merangkum hasil penelitian yang telah dilakukan dan menyajikan kesimpulan yang diperoleh. Selain itu, saran-saran untuk pengembangan lebih lanjut juga diberikan, terutama terkait penelitian masa depan dalam topik yang serupa.

# BAB II
# LANDASAN TEORI

## 2.1 Diabetes Melitus

Kadar glukosa darah yang tinggi merupakan salah satu tanda utama penyakit kronis yang dikenal sebagai diabetes mellitus. Penyakit ini terjadi karena tubuh tidak mampu memproduksi insulin dalam jumlah yang cukup, atau karena tubuh tidak dapat menggunakan insulin secara efektif. Diabetes terdiri dari beberapa jenis, dengan diabetes tipe-1 dan tipe-2 sebagai yang paling umum. Diabetes tipe-1 biasanya disebabkan oleh gangguan autoimun yang merusak sel beta pankreas, sedangkan diabetes tipe-2 sering dikaitkan dengan resistensi insulin akibat gaya hidup yang kurang sehat, seperti obesitas dan kurangnya aktivitas fisik.

Selain itu, ada juga diabetes gestasional yang muncul selama kehamilan dan dapat meningkatkan resiko seseorang terkena diabetes tipe-2 di masa depan. Faktor resiko utama diabetes mencakup genetika, pola hidup yang tidak sehat, obesitas, dan penuaan. Penyakit ini tidak hanya mempengaruhi kadar gula darah, tetapi juga dapat menimbulkan komplikasi serius, seperti neuropati (kerusakan saraf), nefropati (kerusakan ginjal), retinopati (kerusakan retina), serta meningkatkan resiko penyakit jantung dan pembuluh darah.

Data yang disampaikan International Diabetes Federation menunjukkan bahwa jumlah penderita diabetes terus meningkat secara global, dengan Indonesia menjadi salah satu negara dengan jumlah penderita tertinggi. Tren ini terutama disebabkan oleh perubahan gaya hidup di wilayah perkotaan, di mana pola makan tinggi gula dan lemak semakin umum(IDF, 2021).

## 2.2 Pengendalian Konsumsi Gula

Pengendalian konsumsi gula merupakan langkah penting dalam mencegah berbagai resiko kesehatan, terutama diabetes mellitus. Upaya ini didefinisikan sebagai pengaturan asupan gula tambahan agar tidak melebihi batas yang direkomendasikan. Gula tambahan, yang umumnya ditemukan dalam makanan olahan dan minuman manis, dapat memberikan dampak negatif terhadap kesehatan apabila dikonsumsi secara berlebihan. World Health Organization (WHO) merekomendasikan agar konsumsi gula tambahan tidak melebihi 10% dari total kebutuhan energi harian. Untuk memperoleh manfaat kesehatan yang lebih besar, WHO menyarankan pengurangan konsumsi gula hingga di bawah 5% dari total energi harian, yang setara dengan 25 gram atau sekitar enam sendok teh per hari bagi orang dewasa.

Konsumsi gula, garam, dan lemak (GGL) yang melebihi rekomendasi merupakan masalah kesehatan utama di Indonesia. Analisis Survei Konsumsi Makanan Individu menunjukkan bahwa sekitar 29,7% penduduk Indonesia mengkonsumsi gula, garam, dan

lemak melebihi batas yang dianjurkan setiap harinya. Pada kelompok remaja usia 13–18 tahun, proporsi konsumsi gula berlebih bahkan mencapai 53,1% (Masri et al., 2022). Selain itu, data Kementerian Kesehatan RI mencatat bahwa 61,27% penduduk Indonesia usia tiga tahun ke atas mengkonsumsi minuman manis lebih dari satu kali per hari. Konsumsi GGL berlebih ini berkontribusi pada meningkatnya resiko penyakit tidak menular seperti obesitas, diabetes, dan penyakit kardiovaskular (CNN Indonesia, 2022)

## 2.3 Analisis Konsumsi Gula Berbasis Kalori

### 2.3.1 Perhitungan Kebutuhan Kalori Harian

Untuk menentukan batas konsumsi gula harian yang direkomendasikan, langkah pertama yang perlu dilakukan adalah menghitung kebutuhan energi individu. Salah satu metode yang umum digunakan dalam perhitungan kebutuhan kalori adalah rumus Harris-Benedict, yang mempertimbangkan faktor berat badan, tinggi badan, usia, dan tingkat aktivitas fisik(Abrue, 2022; Pavlidou et al., 2023; Takemae et al., 2020) .

Rumus Harris-Benedict:

- Laki-laki = 66 + (13,7 × berat badan) + (5 × tinggi badan) – (6,8 × usia)
- Wanita = 655 + (9,6 × berat badan) + (1,8 × tinggi badan) – (4,7 × usia)

Di mana:

- Berat badan dalam kilogram (kg)
- Tinggi badan dalam centimeter (cm)
- Usia dalam tahun

Setelah kebutuhan energi basal (Basal Metabolic Rate/BMR) diperoleh, nilai ini dikalikan dengan faktor aktivitas untuk mendapatkan kebutuhan kalori total per hari:

- Sangat jarang berolahraga → dikali 1,2
- Jarang olahraga (1-3 kali/minggu) → dikali 1,375
- Cukup olahraga (3-5 kali/minggu) → dikali 1,55
- Sering olahraga (6-7 kali/minggu) → dikali 1,725
- Sangat sering olahraga (sekitar 2 kali sehari) → dikali 1,9

Hasil akhir dari perhitungan ini adalah total kalori yang dibutuhkan individu dalam sehari untuk mempertahankan berat badan ideal.

### 2.3.2 Konversi Kebutuhan Kalori ke Batas Konsumsi Gula Harian

Berdasarkan rekomendasi World Health Organization (WHO), konsumsi gula tambahan tidak boleh melebihi 10% dari total kebutuhan energi harian, Untuk mendapatkan batas

konsumsi gula dalam satuan gram, hasil perhitungan 10% dari total kebutuhan kalori harian tersebut dibagi dengan 4, karena 1 gram gula mengandung 4 kkal.

Rumus perhitungan konsumsi gula harian:

$$\text{Batas Konsumsi gula(gram)} = \frac{10\% \text{ x Total kalori}}{4} \qquad (2.1)$$

## 2.4 Teknologi Pengembangan Aplikasi

### 2.4.1 Android

Android merupakan sistem operasi yang dikembangkan oleh Google khusus untuk perangkat mobile Berbasis kernel Linux, android dirancang untuk memudahkan pengembangan dan penggunaan berbagai aplikasi seperti smartphone dan tablet, baik untuk kebutuhan individu maupun industri (Setiawan, 2020). Dengan kemampuannya sebagai platform yang mendukung pengembangan aplikasi, Android menyediakan lingkungan yang lengkap untuk pengembang, mencakup bahasa pemrograman, framework, dan berbagai alat bantu yang diperlukan.

### 2.4.2 Android Studio

Android Studio adalah lingkungan pengembangan terpadu (Integrated Development Environment atau IDE) yang dirancang secara khusus untuk memfasilitasi proses pembuatan aplikasi Android. Sebagai alat utama yang dikembangkan oleh Google, Android Studio menawarkan sejumlah fitur yang memungkinkan pengembang untuk mengelola seluruh siklus hidup aplikasi dengan efisien mulai dari penulisan kode, pengujian, hingga proses debugging.

IDE ini menyediakan editor kode yang canggih, yang mampu menyoroti sintaks, memberikan saran kode, dan melakukan pemformatan otomatis, sehingga memudahkan pengembang dalam menulis kode program. Bahasa pemrograman yang didukung oleh Android Studio meliputi Java dan Kotlin, dua bahasa yang populer dan banyak digunakan dalam pengembangan aplikasi Android. Selain itu, pengembang dapat dengan mudah membangun antarmuka pengguna (User Interface) melalui alat tata letak visual yang intuitif (Sibuea et al., 2022).

Di samping fitur desain, Android Studio juga memiliki alat debugging yang kuat, memungkinkan pengembang untuk memantau aplikasi secara real-time. Fitur logcat, misalnya, dapat digunakan untuk melacak log aplikasi, membantu menganalisis kinerja, serta menemukan dan mengatasi bug atau kesalahan yang mungkin muncul selama proses

pengembangan. Hal ini memastikan aplikasi berjalan lancar dan memenuhi standar kinerja yang diharapkan sebelum dirilis ke pengguna.

Android Studio juga terintegrasi penuh dengan Android SDK (Software Development Kit), yang menyediakan berbagai pustaka dan komponen penting yang mendukung pengembangan aplikasi Android. Dengan integrasi ini, pengembang dapat dengan mudah mengelola dependensi proyek, memilih versi SDK yang tepat, dan mengakses berbagai alat pengembangan yang diperbarui oleh Google. Semua fitur ini memberikan pengembang fleksibilitas dalam pengembangan aplikasi yang lebih canggih dan interaktif (Hamas & Imaduddin, 2019).

### 2.4.3 Kotlin

Kotlin adalah bahasa pemrograman modern yang diperkenalkan oleh JetBrains pada tahun 2011. Dikenal karena sifatnya yang ramah pengguna, Kotlin dirancang untuk mengatasi tantangan dalam pengembangan aplikasi Android dan telah diadopsi oleh Google sebagai bahasa utama untuk platform tersebut. Dengan fitur-fitur seperti interoperabilitas dengan Java, keamanan tipe (null safety), dan dukungan untuk pemrograman fungsional, Kotlin menawarkan cara yang lebih efisien untuk menulis kode.(Kothari et al., 2024)

Kotlin mendukung pengembangan aplikasi tidak hanya di Android tetapi juga di platform lain, termasuk iOS dan web. Kelebihan Kotlin meliputi sintaks yang ringkas, kemampuan untuk mengurangi boilerplate *Code*, serta fitur-fitur seperti coroutines yang memudahkan pengelolaan tugas asinkron. Dengan demikian, Kotlin menjadi pilihan unggul bagi pengembang yang mencari bahasa pemrograman yang n dan produktif.

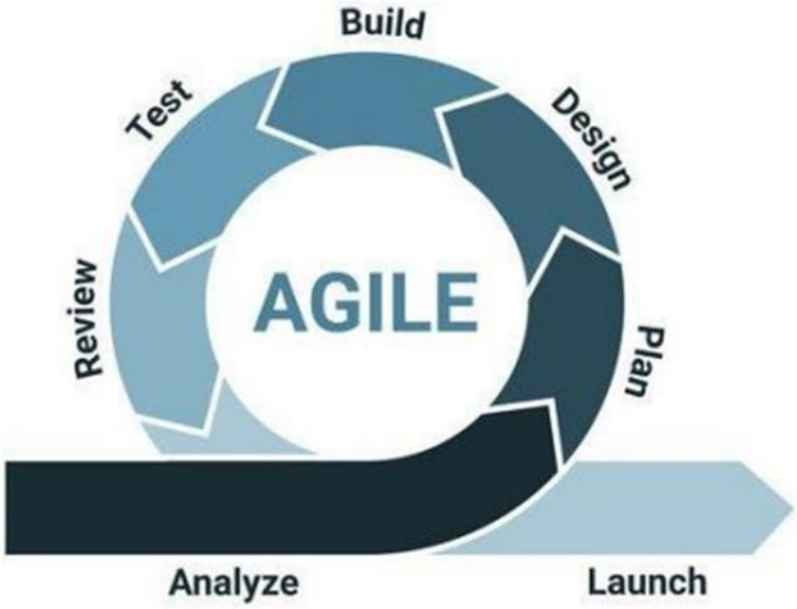
### 2.4.4 Retrofit

Retrofit adalah pustaka yang dikembangkan oleh Square yang berfungsi untuk memudahkan komunikasi antara aplikasi Android dengan server melalui HTTP. Dengan menggunakan Retrofit, pengembang dapat mengakses data dari web service dan mengintegrasikannya langsung ke dalam aplikasi. Pustaka ini mendukung berbagai format data, seperti JSON dan XML, serta dapat dikonfigurasi dengan berbagai metode otentikasi. Retrofit sering digunakan untuk aplikasi yang membutuhkan pertukaran data dengan API eksternal, karena dapat menangani permintaan HTTP (GET, POST, PUT, DELETE) dengan mudah.(Anupam Chugh, 2022)

## 2.5 Agile Development

Pendekatan Agile adalah metodologi pengembangan perangkat lunak yang fleksibel dan iteratif, memungkinkan pengembang untuk merespons perubahan dengan cepat dan melibatkan

pengguna secara aktif dalam proses pengembangan. Dalam konteks aplikasi pemantauan konsumsi gula harian, metodologi Agile memungkinkan tim pengembang untuk terus memperbaiki aplikasi berdasarkan umpan balik pengguna secara berkelanjutan.Proses Agile terdiri dari beberapa tahapan kunci sebagai berikut:




a. Analisis:

   Analisis kebutuhan adalah tahapan awal dalam metode Agile . Proses awal tahap ini adalah melakukan pengumpulan data-data sebagai bahan informasi untuk pengembangan sistem. Pengumpulan data dilakukan dengan teknik observasi atau wawancara. Tahap analisis kebutuhan menghasilkan Informasi yang diperoleh digunakan untuk menyusun backlog kebutuhan, mengidentifikasi fitur utama, dan mendefinisikan tujuan pengembangan aplikasi.

b. Perencanaan (*Plan*):

   Pada tahap ini, tim mengidentifikasi pekerjaan yang harus diselesaikan dalam setiap iterasi (sprint) dan menyusun rencana pengembangan. *Backlog* fitur yang dihasilkan dari tahap analisis diprioritaskan berdasarkan kebutuhan pengguna. Setiap sprint memiliki tujuan spesifik yang ditetapkan untuk memastikan kemajuan pengembangan dapat diukur secara akurat. Perencanaan ini juga mencakup penjadwalan, dan penetapan indikator keberhasilan.

c. Perancangan Desain:

   Pada tahap perancangan desain, struktur sistem dan antarmuka pengguna (UI) mulai dirancang. Desain ini mencakup pembuatan *Wireframe*, *Use Case Diagram*, dan

*Activity Diagram*. *Wireframe* memberikan gambaran awal tata letak antarmuka aplikasi, sementara *Use Case Diagram* menunjukkan interaksi antara pengguna dan sistem. *Activity Diagram* digunakan untuk memvisualisasikan alur proses aplikasi secara keseluruhan.

d. Pengembangan:

Tahap pengembangan merupakan proses implementasi fitur-fitur aplikasi berdasarkan backlog yang telah disusun. Pengembangan dilakukan menggunakan iterasi pendek (sprint), di mana setiap sprint fokus pada penyelesaian satu atau beberapa fitur utama. Dalam proyek ini, bahasa pemrograman Kotlin dan framework Android SDK digunakan untuk mengembangkan aplikasi. Proses ini juga mencakup integrasi backend dengan menggunakan API dan penyimpanan data.

e. Pengujian:

Pada tahap ini, hasil implementasi kode pemrograman akan melalui proses pengujian. Proses ini bertujuan untuk menguji kesesuaian antara output sistem dengan kebutuhan yang telah dirancang sebelumnya. Pengujian dilakukan setelah setiap iterasi selesai untuk memastikan kualitas aplikasi. Metode Black Box Testing digunakan untuk mengevaluasi apakah fitur yang dikembangkan telah memenuhi spesifikasi fungsional. Selain itu, pengujian ini bertujuan untuk menemukan bug atau kesalahan dalam aplikasi agar dapat segera diperbaiki sebelum melanjutkan ke iterasi berikutnya. Tidak hanya Black Box Testing, tetapi pengujian juga dilakukan menggunakan metode System Usability Scale (SUS) untuk menilai tingkat kegunaan aplikasi berdasarkan pengalaman pengguna. Proses pengujian yang konsisten memastikan bahwa aplikasi berjalan dengan baik dan memberikan pengalaman pengguna yang optimal.

## 2.6 Perbandingan Metode Pengembangan

Metode Agile adalah pendekatan yang fleksibel dan adaptif, yang menekankan pada siklus iteratif dan kolaborasi antar tim lintas fungsi. Pendekatan ini memungkinkan respons cepat terhadap perubahan kebutuhan pengguna dan persyaratan proyek, serta memastikan perbaikan berkelanjutan dalam pengembangan produk(Rahmaliani et al., 2023). Agile memungkinkan tim untuk bekerja secara dinamis dan cepat merespons perubahan, dengan melibatkan berbagai pihak dalam setiap iterasi proyek. Hal ini sangat bergantung pada komunikasi terbuka dan kerjasama yang kuat antar anggota tim, yang memungkinkan identifikasi masalah dan solusi dilakukan secara cepat dan tepat waktu. Pendekatan ini sangat

bermanfaat untuk proyek pengembangan perangkat lunak yang memerlukan pembaruan terus-menerus dan adaptasi cepat terhadap perubahan prioritas, karena sifatnya yang mengutamakan fleksibilitas dan kecepatan dalam mengambil keputusan(Hasan et al., 2021).

Model Pengembangan Waterfall adalah salah satu metode pengembangan perangkat lunak yang paling tua dan banyak digunakan. Metode ini bersifat linier, di mana setiap tahap dalam pengembangan perangkat lunak harus selesai terlebih dahulu sebelum melanjutkan ke tahap berikutnya(Hasan et al., 2021). Tahapan yang umum dalam model Waterfall meliputi analisis kebutuhan, perancangan sistem, implementasi, pengujian, dan pemeliharaan. Setiap tahap dilakukan dengan urutan yang ketat, dan tidak ada ruang untuk kembali ke tahap sebelumnya setelah langkah tersebut diselesaikan. Pendekatan ini mengutamakan dokumentasi yang lengkap di setiap tahap, yang memungkinkan pengelolaan proyek menjadi lebih terstruktur dan jelas. Namun, model Waterfall kurang fleksibel terhadap perubahan dan sering kali tidak cocok untuk proyek yang memiliki kebutuhan yang dinamis atau kompleks, yang cenderung berubah selama proses pengembangan.

Metode Spiral adalah pendekatan pengembangan perangkat lunak yang menggabungkan elemen-elemen dari metode Waterfall dan Prototyping. Pendekatan ini dikenal sebagai model evolusioner yang memungkinkan proses pengembangan perangkat lunak dilakukan secara iteratif dan terkontrol, dimulai dengan perencanaan yang jelas, namun dengan fleksibilitas untuk menambahkan perubahan yang diperlukan selama proses berlangsung(Gunawan et al., 2023). Model ini dibangun dalam beberapa spiral atau putaran, di mana setiap spiral meliputi langkah-langkah seperti perencanaan, identifikasi resiko, pengembangan dan pengujian, serta evaluasi oleh pengguna. Salah satu keuntungan utama dari model Spiral adalah kemampuannya untuk mengelola resiko dan memastikan setiap fase sistem dapat dievaluasi dengan umpan balik yang didapatkan dari pengguna atau klien. Spiral cocok digunakan untuk proyek yang membutuhkan perubahan atau pengembangan bertahap, terutama ketika kebutuhan atau spesifikasi sistem dapat berkembang seiring waktu.

| No | Metode | Penulis | Maksud Penelitian | Kelebihan | kekurangan |
|---|---|---|---|---|---|
| 1 | Agile | (Hikmah et al., 2021) | Penggunaan Metode Agile Untuk Meningkatkan Kreativitas Guru Melalui Berbagi Pengetahuan | • Fleksibilitas dalam menghadapi perubahan | • Memerlukan keterlibatan pengguna yang tinggi |

| | | | | | |
|---|---|---|---|---|---|
| | | | (Knowledge Sharing) (Studi Kasus: SDN Cipulir 03 Kebayoran Lama, Jakarta) | • Keterlibatan pengguna yang lebih aktif<br>• Pengiriman produk lebih cepat | • Kesulitan dalam perencanaan jangka panjang<br>• Memerlukan keterampilan manajemen proyek yang kuat |
| 2 | Agile | (Rahmaliani et al., 2023) | Merancang dan mengimplementasikan sistem informasi pelaporan mortalitas pasien rawat inap dengan metode Agile, guna meningkatkan efisiensi dan keakuratan pelaporan mortalitas | • Pengembangan sistem lebih cepat<br>• Kemudahan dalam adaptasi terhadap perubahan kebutuhan<br>• Meningkatkan kualitas dan efisiensi pelaporan | • Diperlukan evaluasi yang berulang-ulang yang bisa mempengaruhi jadwal proyek<br>• Komunikasi intensif yang dibutuhkan dapat menjadi tantangan di tim besar |
| 3 | Waterfall | (Deva Purwanto & Zukhri, 2023) | Pengembangan sistem informasi untuk perumahan yang menggunakan metode Waterfall untuk mengelola | • Struktur yang jelas dan mudah diikuti<br>• Tahapan yang terurut | • Tidak fleksibel terhadap perubahan kebutuhan |

| | | | | | |
|---|---|---|---|---|---|
| | | | data penghuni, pembayaran, dan pengelolaan kegiatan sosial dengan fokus pada pengelolaan pendataan perumahan secara efisien dan efektif. | membuatnya mudah dikelola dan diterapkan<br>• Cocok untuk proyek dengan persyaratan yang jelas dan stabil | • Setiap tahapan tidak dapat kembali ke tahap sebelumnya<br>• Mengambil waktu lebih lama untuk proyek besar karena pengujian dan validasi yang terpisah dalam setiap tahap |
| 4 | Waterfall | (Hasan et al., 2021) | Pembangunan sistem informasi untuk pemantauan dan evaluasi pelayanan kesehatan berbasis web menggunakan metode Waterfall untuk menangani data dan pelaporan yang lebih terstruktur dan efisien. | • Mendukung dokumentasi yang lengkap dan terstruktur<br>• Dapat menghasilkan sistem yang stabil jika persyaratannya jelas dari awal<br>• Memudahkan pelaksanaan | • Membutuhkan komitmen waktu yang Panjang<br>• Tidak dapat beradaptasi dengan mudah jika ada perubahan mendadak dalam spesifikasi sistem yang diinginkan |

| | | | | | |
|---|---|---|---|---|---|
| | | | | pengujian karena pengujian dilakukan pada setiap tahapan | |
| 5 | Spiral | (Gunawan et al., 2023) | Pengembangan sistem informasi aplikasi Puskesmas untuk memantau kecukupan gizi ibu hamil dan menyusui. Sistem dikembangkan dengan pendekatan Spiral untuk memberikan fleksibilitas dalam memenuhi kebutuhan pengguna dan memperbaiki sistem berdasarkan feedback. | • Memungkinkan perubahan dan penyesuaian yang cepat selama pengembangan<br>• Mampu menangani resiko dengan lebih baik<br>• Menghasilkan perangkat lunak yang dapat lebih mudah beradaptasi dengan kebutuhan pengguna | • Membutuhkan sumber daya lebih besar dan biaya yang lebih tinggi<br>• Proses yang lebih lama karena banyaknya iterasi dan evaluasi yang dilakukan. |

| | | | | | |
|---|---|---|---|---|---|
| 6 | Spiral | (Mulyanto et al., 2020) | Pengembangan sistem monitoring perkembangan anak di TK IT TAAMASA menggunakan metode Spiral untuk memberikan solusi yang lebih fleksibel dalam memantau perkembangan anak secara online dan real-time. | • Memberikan fleksibilitas untuk menyesuaikan sistem dengan kebutuhan yang berkembang<br>• Mengoptimalkan interaksi antara pengembang dan pengguna (orang tua dan guru) dengan melibatkan feedback secara teratur | • Bisa memakan waktu lebih lama dan biaya yang lebih tinggi, karena setiap iterasi memerlukan pengujian dan evaluasi kembali. |

Berdasarkan Tabel diatas yang membandingkan berbagai pendekatan dalam pengembangan aplikasi, pendekatan Agile dipilih sebagai metode utama dalam pengembangan aplikasi bergerak Memantau Konsumsi Gula Harian. Pendekatan ini dipandang sebagai pilihan terbaik karena memungkinkan penanganan perubahan kebutuhan dengan cepat, melibatkan pengguna secara aktif, dan memungkinkan pengiriman produk secara iteratif. Dalam konteks pengembangan aplikasi Android untuk memantau konsumsi gula harian dalam mencegah resiko diabetes, penggunaan Agile akan memungkinkan pengembangan berkelanjutan yang

responsif terhadap kebutuhan dan umpan balik pengguna, yang sangat penting mengingat sifat dinamis dari informasi kesehatan dan kebiasaan makan yang dapat bervariasi antar pengguna.

Selain itu, kolaborasi dengan rekan peneliti dalam pengembangan model deteksi makanan berdasarkan gambar menjadi salah satu alasan penting untuk memilih metode Agile. Kolaborasi ini memungkinkan integrasi komponen baru secara bertahap dan fleksibel, di mana setiap iterasi Agile dapat mencakup perbaikan dan pembaruan model deteksi tersebut berdasarkan umpan balik dari pengujian pengguna. Metode Agile memungkinkan tim untuk terus menyesuaikan dan menyempurnakan fitur deteksi makanan dalam aplikasi, memastikan bahwa aplikasi dapat memberikan hasil yang lebih akurat dan relevan dengan perkembangan teknologi serta kebutuhan pengguna.

## 2.7 Tinjauan Pustaka Fitur Aplikasi

Penelitian ini mengacu pada berbagai studi terdahulu yang relevan dengan pengembangan aplikasi pemantauan konsumsi gula harian berbasis Android. Kajian pustaka mencakup penelitian tentang pemantauan kesehatan, penggunaan teknologi mobile, dan fitur interaktif untuk meningkatkan keterlibatan pengguna. Hasil kajian menunjukkan berbagai pendekatan yang dapat menjadi acuan dalam merancang sistem yang efektif dan inovatif.

Penelitian pertama dilakukan oleh (Suwan et al., 2022) yang mengembangkan aplikasi berbasis mobile dengan fitur rekomendasi personalisasi berbasis konteks untuk memantau konsumsi gula pada remaja di Thailand. Teknologi yang digunakan mencakup pengolahan data pribadi, kesehatan, dan gaya hidup pengguna untuk menghasilkan rekomendasi otomatis yang relevan. Fitur utama aplikasi ini meliputi laporan konsumsi gula, grafik statistik, profil pengguna, dan pengaturan preferensi. Penelitian ini memberikan wawasan penting tentang pentingnya fitur personalisasi dalam membantu pengguna memantau konsumsi gula secara efektif.

Penelitian kedua oleh (Beun et al., 2021) mengembangkan aplikasi gamifikasi e-coaching untuk membantu remaja mengurangi konsumsi minuman manis (SSB). Fitur teknis dalam aplikasi ini mencakup avatar personalisasi, tantangan gamifikasi, penghargaan, serta laporan statistik. Hasil penelitian menunjukkan bahwa elemen gamifikasi berhasil meningkatkan keterlibatan pengguna. Penelitian ini menjadi acuan dalam merancang elemen motivasi dan penghargaan untuk meningkatkan pengalaman pengguna dalam memantau konsumsi gula harian.

Penelitian ketiga dilakukan oleh (Plecher et al., 2020)yang merancang aplikasi TrackSugAR berbasis iOS untuk memantau konsumsi gula menggunakan teknologi

Augmented Reality (AR). Aplikasi ini dilengkapi dengan fitur seperti visualisasi gula dalam bentuk kubus 3D, pemindaian bar*Code* makanan, dan pelacakan data melalui grafik. Teknologi AR yang digunakan memberikan cara interaktif untuk meningkatkan kesadaran pengguna tentang konsumsi gula mereka. Penelitian ini menjadi inspirasi dalam mengembangkan fitur visualisasi data yang intuitif dan menarik.

Penelitian keempat oleh (Ding, 2019)mengembangkan aplikasi mobile bernama Smart Tracking for Healthier Lifestyle untuk memantau konsumsi gula harian. Aplikasi ini menggunakan teknologi pemindaian bar*Code* untuk merekam data makanan, visualisasi grafik konsumsi gula, dan pengingat batas konsumsi harian. Hasil evaluasi menunjukkan bahwa aplikasi ini mudah digunakan dan membantu pengguna meningkatkan kesadaran mereka terhadap pola makan. Studi ini mendukung pengembangan fitur pengingat dan laporan harian yang membantu pengguna memahami konsumsi gula mereka secara lebih mendalam.

Penelitian kelima oleh (Sari et al., 2022) mengembangkan aplikasi mobile untuk monitoring kalori harian untuk mencegah penyakit diabetes. Aplikasi ini memiliki tampilan halaman login, tampilan home berupa edukasi, tampilan halaman perhitungan rekomendasi kalori halaman catatan makanan. Hasil pengujian menggunakan usability menunjukkan bahwa aplikasi ini berhasil menunjukan tigakat yang baik dengan 92% responden menilai dapat diterima.

Penelitian keenam oleh(Zuhri et al., 2024) mengembangkan aplikasi mobile untuk monitoring kadar gula darah. Aplikasi ini memiliki tampilan monitoring gula darah yang mencakup mencakup grafik gula darah, pencatatan aktivitas, dan alarm sebagai faktor penting dalam manajemen diabetes. Fitur perawatan diabetes berisi rekomendasi-rekomendasi yang bersumber dari para pakar untuk mengedukasi dan mendorong pasien agar menerapkan gaya hidup yang sehat yang disesuaikan dengan kondisi terkini dari pasien diabetes.

| No | Fitur Aplikasi Kamu | Suwan et al., 2022 | Beun et al., 2021 | Plecher et al., 2020 | Ding, 2019 | Sari et al., 2022 | Zuhri et al., 2024 |
|---|---|---|---|---|---|---|---|
| 1 | Informasi Pengguna (Kalori & Gula) | ✓ | ✓ | - | - | ✓ | ✓ |
| 2 | Pencatatan Konsumsi Gula (Manual & Otomatis) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 3 | Notifikasi Konsumsi Gula | - | - | - | ✓ | - | ✓ |

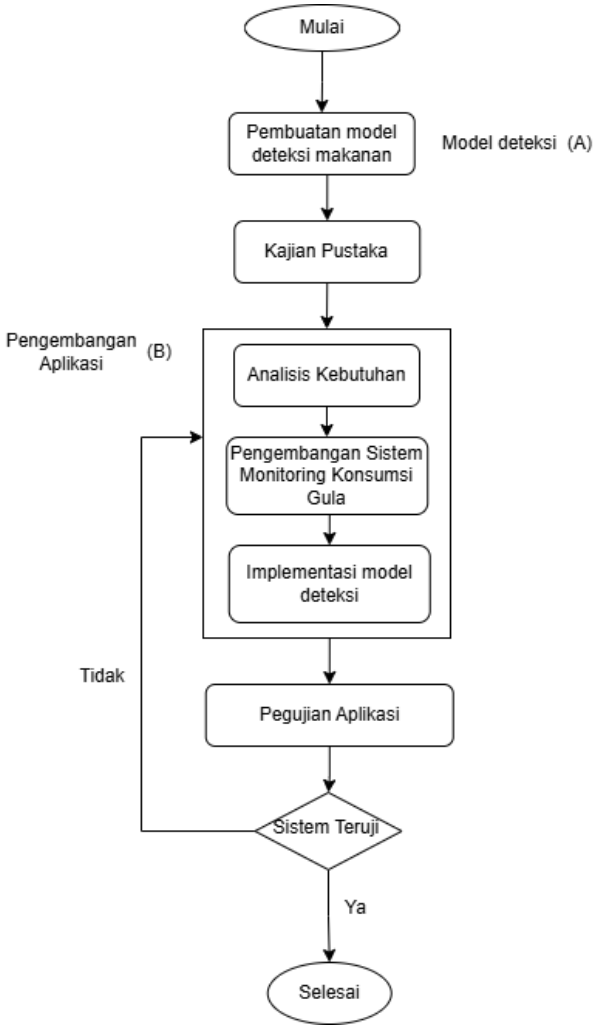
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 4 | Edukasi (Artikel dan Tips) | - | - | - | - | ✓ | ✓ |
| 5 | Monitoring & Laporan Statistik | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6 | Chatbot untuk Tanya Jawab Kesehatan | ✓ | - | - | - | - | - |
| 7 | Panduan Konsumsi Gula & Status Pengguna | - | - | - | - | - | ✓ |
| 8 | Scan Barcode Produk | - | - | ✓ | ✓ | - | - |
| 9 | Fitur Login, Register, Profile | ✓ | ✓ | - | - | ✓ | ✓ |

Berdasarkan hasil analisis perbandingan dari penelitian terdahulu, fitur utama aplikasi pemantauan konsumsi gula meliputi rekomendasi yang dipersonalisasi, pelacakan konsumsi gula dalam bentuk grafik, laporan harian, dan elemen gamifikasi untuk meningkatkan keterlibatan pengguna. Namun, kebaruan yang ditawarkan oleh penelitian ini terletak pada pengembangan fitur notifikasi yang lebih proaktif dan personal.

Fitur notifikasi ini mencakup pengingat sebelum konsumsi gula mencapai batas harian, notifikasi edukatif dengan saran praktis untuk membantu pengguna membuat keputusan makanan yang lebih sehat, serta notifikasi berbasis waktu yang relevan dengan aktivitas pengguna. Dengan pendekatan ini, aplikasi tidak hanya membantu pengguna dalam memantau dan mengontrol konsumsi gula, tetapi juga mendorong terciptanya kebiasaan pola makan yang lebih sehat secara berkelanjutan.

# BAB III
# METODOLOGI

Proses penelitian ini mengadopsi suatu metodologi untuk memastikan pengidentifikasian langkah-langkah secara sistematis, yang nantinya dapat berfungsi sebagai panduan terstruktur dalam menangani permasalahan yang dihadapi. Rinciannya dapat diamati melalui representasi visual pada Gambar 3. 1




Gambar 3. 1 Proses pembuatan aplikasi dan implementasi model

Gambar 3.1 menggambarkan alur metodologi yang diterapkan dalam penelitian ini. Pada bagian A, pembuatan model deteksi makanan dilakukan oleh rekan peneliti, di mana model machine learning dikembangkan untuk mendeteksi kandungan gula dalam makanan. Setelah model selesai, fokus penelitian saya beralih ke B, yang meliputi pengembangan aplikasi untuk pemantauan konsumsi gula. Pada bagian ini, saya memfokuskan pada analisis kebutuhan, pengembangan sistem monitoring konsumsi gula, serta implementasi model deteksi yang telah dibuat. Setelah aplikasi dan model diterapkan, pengujian dilakukan untuk memastikan

fungsionalitas dan keandalan sistem dalam membantu pengguna memantau konsumsi gula mereka secara otomatis.

### 3.1 Model Deteksi AI

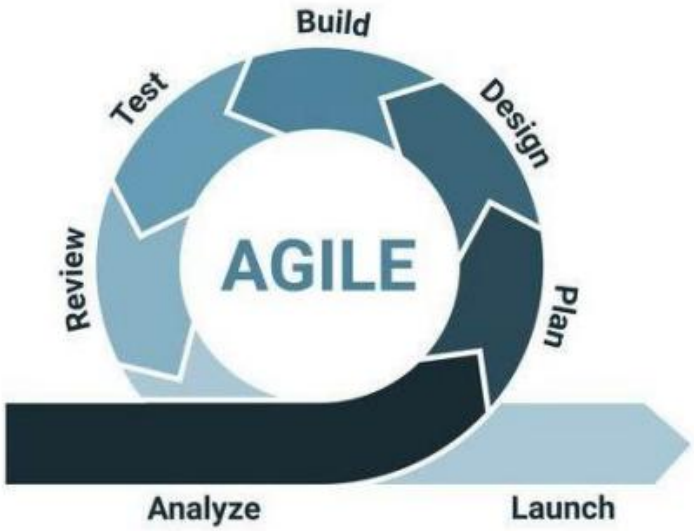
Pada tahap pembuatan model deteksi makanan, sistem ini mengintegrasikan model machine learning yang telah dikembangkan oleh rekan peneliti. Model tersebut dirancang untuk mengidentifikasi kandungan gula dalam makanan berdasarkan citra visual atau deskripsi teks produk. Dalam pengembangannya, Rafli menggunakan pendekatan Convolutional Neural Network (CNN) dengan arsitektur EfficientNetB7, yang dikombinasikan dengan teknik transfer learning guna meningkatkan efisiensi pelatihan dan akurasi prediksi.

Dataset yang digunakan merupakan hasil seleksi dari Food-101, difokuskan pada makanan yang relevan dengan konteks konsumsi masyarakat Indonesia, serta diperkaya dengan estimasi kandungan gula pada setiap kelas makanan. Proses pelatihan model mencakup augmentasi data untuk meningkatkan kemampuan generalisasi serta optimisasi beberapa parameter penting seperti *learning rate*, *batch size*, dan jumlah *epoch* untuk mengurangi resiko *overfitting*.

Hasil pelatihan menunjukkan kinerja model yang sangat baik, dengan rata-rata akurasi mencapai sekitar 87% setelah dilakukan transformasi ulang terhadap data uji. Peningkatan akurasi ini terutama terlihat pada kelas-kelas makanan yang sebelumnya sulit dikenali.

Karena model deteksi ini berasal dari penelitian terpisah, fokus utama sistem bukan pada pelatihan ulang, melainkan pada integrasi dan implementasi model tersebut ke dalam aplikasi pemantauan konsumsi gula. Model ini kemudian diadaptasi untuk memungkinkan pengguna mengenali kadar gula dalam makanan secara otomatis, baik melalui pengambilan gambar maupun input berbasis teks. Proses integrasi ini menjadi salah satu komponen penting dalam fitur utama aplikasi GlucoFit, yang bertujuan membantu pengguna dalam pengelolaan konsumsi gula harian secara lebih cerdas dan informatif.

## 3.2 Pegembangan Aplikasi




Gambar 3. 2 Metodelogi Agile

Proses pengembangan aplikasi menggunakan pendekatan Agile ini dibagi menjadi beberapa tahapan utama, yaitu Analisis, Perencanaan plan, Perancangan desain, Pengembangan, Pengujian, Tinjauan, dan Peluncuran. Setiap tahapannya dilakukan dalam siklus iterasi yang berkelanjutan, memungkinkan pengembang untuk melakukan penyesuaian di setiap tahap pengembangan berdasarkan hasil evaluasi di sprint sebelumnya

### 3.2.1 Tahap Analisis (Analysis)

Tahap pertama dari metode Agile, yaitu Analisis (Analyze), merupakan langkah fundamental dalam proses pengembangan produk atau layanan. Pada tahap ini, pengembang memfokuskan upaya pada pengumpulan dan pemahaman kebutuhan pengguna. Proses ini sangat penting karena menentukan arah keseluruhan pengembangan aplikasi, memastikan bahwa solusi yang dikembangkan benar-benar sesuai dengan masalah yang dihadapi pengguna. Agar proses ini efektif, pengembang melakukan observasi langsung serta mengadakan interaksi dengan calon pengguna, bertujuan untuk mendapatkan gambaran yang jelas tentang tantangan dan harapan pengguna.

Setelah mendapatkan wawasan awal, analisis kebutuhan dilakukan sebagai bagian dari pendekatan sistematis untuk memahami secara detail apa yang diinginkan dan dibutuhkan oleh pengguna. Langkah ini penting untuk mengidentifikasi kebutuhan yang eksplisit maupun tersembunyi, sehingga setiap aspek aplikasi dapat dikembangkan sesuai dengan harapan pengguna. Selain itu, analisis ini memberikan struktur dan prioritas terhadap fitur-fitur yang akan dikembangkan, memastikan bahwa fokus pengembangan tetap pada solusi yang paling berdampak.

Untuk mendukung proses analisis ini, beberapa pertanyaan telah disiapkan untuk mendapatkan wawasan mendalam mengenai kebutuhan dan preferensi pengguna pada tabel 3.1 Berikut adalah pertanyaan yang diajukan kepada pengguna:

Tabel 3. 1 Daftar Pertanyaan Kepada Calon Pengguna

| No. | Pertanyaan | Tipe Pertanyaan | Kategori |
|---|---|---|---|
| 1 | Usia | Isian | Demografi |
| 2 | Jenis Kelamin | Pilihan Ganda Tunggal | Demografi |
| 3 | Apakah Anda memiliki kondisi kesehatan tertentu (misalnya, diabetes)? | Pilihan Ganda Tunggal | Kondisi Kesehatan |
| 4 | Apakah Anda pernah menggunakan aplikasi untuk memantau kesehatan Anda sebelumnya? | Pilihan Ganda Tunggal | Pengalaman Aplikasi |
| 5 | Apa tujuan Anda untuk memantau konsumsi gula harian Anda? | Pilihan Ganda Jamak | Motivasi Pemantauan |
| 6 | Apakah Anda memiliki target konsumsi gula harian tertentu? | Pilihan Ganda Tunggal | Motivasi Pemantauan |
| 7 | Seberapa sering Anda memperhatikan konsumsi gula harian Anda? | Skala Likert | Kebiasaan |
| 8 | Bagaimana biasanya Anda mengetahui jumlah gula yang dikonsumsi? | Pilihan Ganda Jamak | Kebiasaan |
| 9 | Apakah Anda lebih suka mencatat konsumsi gula secara manual atau otomatis? | Pilihan Ganda Tunggal | Preferensi Fitur |
| 10 | Seberapa penting bagi Anda mendapatkan notifikasi saat gula mendekati batas harian? | Skala Likert | Preferensi Fitur |
| 11 | Apakah Anda ingin aplikasi memberi saran/peringatan saat gula melewati batas? | Skala Likert | Preferensi Fitur |
| 12 | Apakah fitur riwayat konsumsi gula harian menarik bagi Anda? | Skala Likert | Preferensi Fitur |
| 13 | Apakah aplikasi perlu menyediakan artikel dan tips pola makan sehat dan pengelolaan gula? | Skala Likert | Preferensi Fitur |

### 3.2.2 Tahap Perencanaan (Plan)

Dalam tahap perencanaan (plan), langkah-langkah yang akan diambil dalam pengembangan direncanakan dan diatur dengan rinci. Proses ini dimulai dengan mengidentifikasi serta merinci tugas-tugas yang perlu diselesaikan dalam bentuk sprint atau

iterasi. Setiap tugas diprioritaskan berdasarkan nilai yang paling penting dan berdampak tinggi. Setelah itu, tujuan jangka pendek ditetapkan untuk dicapai selama setiap sprint atau iterasi. Tujuan-tujuan ini menjadi panduan dalam mengarahkan pekerjaan dan mengukur keberhasilan pengembangan, yang didasarkan pada hasil analisis kebutuhan pengguna yang telah dilakukan pada tahap sebelumnya (Raharjo et al., 2023).

Setelah analisis kebutuhan selesai, perencanaan pengembangan fitur dilakukan berdasarkan prioritas yang telah ditetapkan. Fitur-fitur yang teridentifikasi kemudian disusun dalam backlog pengembangan dan dibagi ke dalam sprint dengan durasi sesuai dengan kebutuhan pegembangan (Deri et al., 2023). Pendekatan Agile memungkinkan evaluasi hasil pengembangan dilakukan pada setiap sprint, sehingga setiap fitur yang dikembangkan dapat diperbaiki sebelum melanjutkan ke tahap berikutnya (Aldisa & Maulana, 2022).

### 3.2.3 Tahap Perancangan (Design)

Pada tahap ini, proses perancangan desain dilakukan untuk menentukan struktur sistem dan alur interaksi pengguna. Perancangan ini mencakup Rancangan Proses Bisnis, Rancangan Basis Data, Rancangan Antar Muka untuk menggambarkan bagaimana pengguna akan berinteraksi dengan aplikasi dan bagaimana alur proses berjalan dalam sistem.

**A. Use Case Diagram**

*Use Case Diagram* digunakan untuk menggambarkan interaksi antara aktor dan sistem dalam memenuhi suatu tujuan. Diagram ini berfungsi untuk mengidentifikasi kebutuhan fungsional dari sebuah sistem serta menggambarkan skenario yang berbeda dari interaksi pengguna dengan sistem. Dalam proses pengembangan perangkat lunak, *Use Case Diagram* membantu tim pengembang dalam memahami alur kerja dari perspektif pengguna

**B. Rancangan basis Data**

Rancangan basis data merupakan proses perencanaan dan penyusunan struktur data yang akan digunakan oleh aplikasi.

**C. Activity Diagram**

*Activity Diagram* digunakan untuk memodelkan alur aktivitas yang terjadi dalam suatu sistem, menggambarkan proses yang kompleks dengan lebih rinci. Diagram ini menunjukkan bagaimana sebuah use case dijalankan, mulai dari aktivitas awal hingga aktivitas akhir. *Activity Diagram* mempermudah pengembang dan analis untuk memahami logika alur kerja dari berbagai aktivitas, sehingga memastikan kesesuaian antara desain sistem dan kebutuhan fungsional yang telah ditentukan

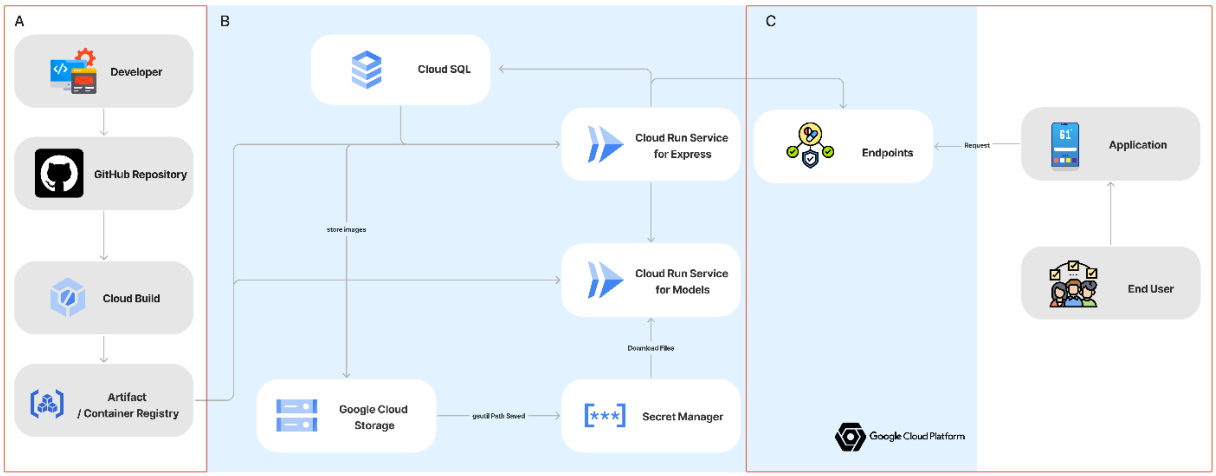
### D. Wireframe

*Wireframe* digunakan pada tahap awal pengembangan untuk merancang antarmuka pengguna (UI) secara fungsional. Fokus utama *Wireframe* adalah tata letak elemen-elemen penting seperti tombol, menu, gambar, dan teks, tanpa melibatkan desain visual atau estetika yang mendetail. Tujuannya adalah membantu memvisualisasikan alur interaksi pengguna dengan aplikasi atau website secara jelas. Prototipe awal dibuat untuk diuji dan disesuaikan sebelum masuk ke tahap pengembangan lebih lanjut. (Deri et al., 2023).

## 3.2.4 Tahap Pengembangan (Build)

### *A. Cloud Architecture*

*Cloud architecture* menjadi pondasi utama dalam pengembangan aplikasi modern karena menawarkan fleksibilitas, skalabilitas, serta kemudahan integrasi dengan berbagai layanan. Google Cloud Platform (GCP) menyediakan berbagai layanan yang mendukung proses pengembangan, penyimpanan, dan pengelolaan data secara efisien. Laporan ini membahas arsitektur cloud yang dirancang menggunakan layanan GCP, meliputi Cloud Run, Cloud SQL, Cloud Storage, Secret Manager, serta mekanisme deployment otomatis melalui Cloud Build, sebagaimana ditunjukkan pada Gambar 3.3.




Gambar 3. 3 *Cloud Architecture*

*Cloud architecture* ini terdiri dari beberapa komponen utama yang saling terhubung dalam proses deployment dan operasional aplikasi, yang dapat dibagi ke dalam tiga bagian: A, B, dan C.

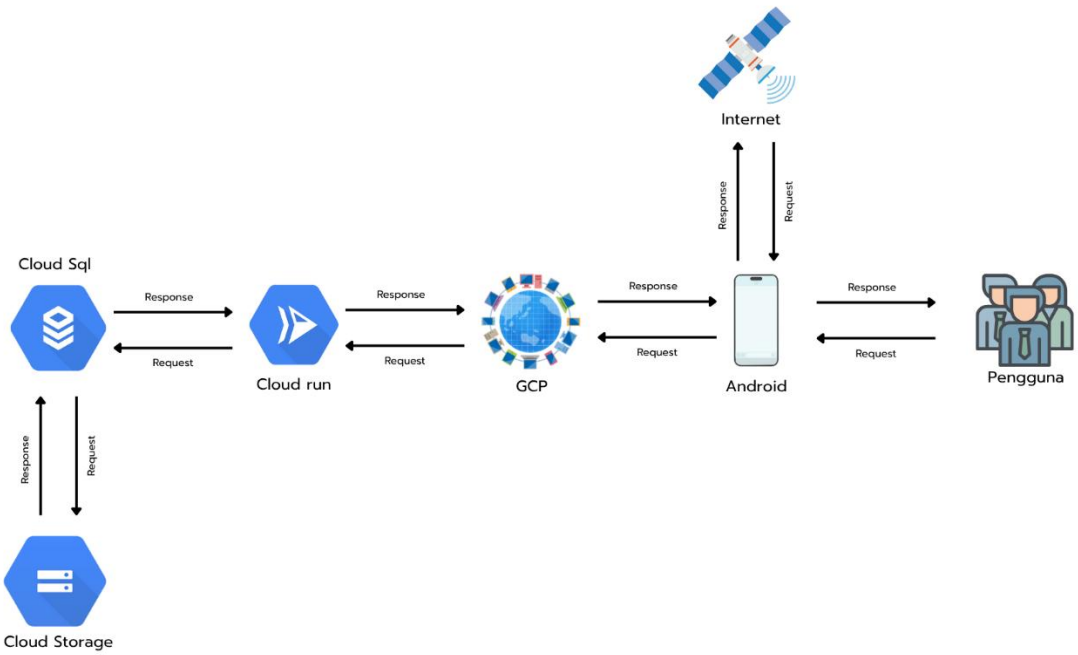
Pada bagian A, Developer mengunggah kode sumber ke GitHub Repository, yang kemudian diproses oleh Cloud Build untuk membangun aplikasi secara otomatis setelah ada perubahan kode. Hasil build aplikasi disimpan dalam bentuk container di Artifact / Container

Registry untuk memastikan ketersediaan dan pengelolaan versi aplikasi yang siap untuk deployment.

Pada bagian B, terdapat dua layanan utama yang dijalankan menggunakan Cloud Run. Cloud Run Service for Express menangani backend aplikasi berbasis Express.js untuk memproses permintaan pengguna, sedangkan Cloud Run Service for Models menjalankan model machine learning atau komputasi khusus lainnya yang diperlukan oleh aplikasi. Data pengguna dan operasional aplikasi disimpan dalam Cloud SQL, sementara file atau model yang dibutuhkan aplikasi disimpan di Google Cloud Storage. Untuk menjaga keamanan, Secret Manager digunakan untuk menyimpan informasi sensitif seperti kredensial atau token API yang digunakan dalam layanan backend.

Pada bagian C, Endpoints berfungsi sebagai API Gateway yang mengelola permintaan dari aplikasi sebelum diteruskan ke backend untuk diproses. Aplikasi yang dikembangkan kemudian digunakan oleh End User untuk berbagai keperluan, seperti input data, analisis, atau pengambilan keputusan.

**B. Arsitektur Sistem**




Gambar 3. 4 Arsitektur Sistem

Gambar 3.4 menunjukkan arsitektur sistem aplikasi Android yang terhubung dengan layanan cloud menggunakan platform Google Cloud Platform (GCP). Sistem ini dirancang untuk memastikan pengelolaan data pengguna secara efisien, aman, dan terpusat. Pengguna berinteraksi langsung melalui aplikasi Android, yang menjadi antarmuka utama untuk

mengakses berbagai fitur seperti pencatatan konsumsi, laporan harian, hingga artikel edukatif. Setiap permintaan dari aplikasi Android dikirim melalui jaringan internet menuju server backend yang di-deploy menggunakan layanan Cloud Run milik GCP.

Cloud Run bertugas memproses permintaan dari aplikasi, menjalankan logika bisnis, dan menghubungkan ke dua layanan utama, yaitu Cloud SQL dan Cloud Storage. Cloud SQL digunakan sebagai basis data relasional untuk menyimpan informasi penting seperti data pengguna, histori konsumsi, dan laporan. Setiap kali aplikasi memerlukan data, Cloud Run akan mengirim permintaan ke Cloud SQL, lalu menerima dan mengolah respons yang diberikan. Di sisi lain, Cloud Storage berfungsi sebagai media penyimpanan untuk berkas-berkas seperti gambar makanan. Cloud Run juga bertanggung jawab melakukan proses upload dari Cloud Storage sesuai kebutuhan sistem.

Secara keseluruhan, arsitektur ini menggambarkan alur komunikasi dua arah antara pengguna dan sistem backend berbasis cloud. Aplikasi Android bertindak sebagai klien, sementara layanan backend yang berjalan di Cloud Run berperan sebagai penghubung antara frontend dan layanan penyimpanan data di cloud. Dengan pendekatan ini, sistem diharapkan mampu memberikan pengalaman pengguna yang responsif, andal, dan aman melalui integrasi penuh dengan layanan GCP.

## C. Arsitektur Perangkat Lunak

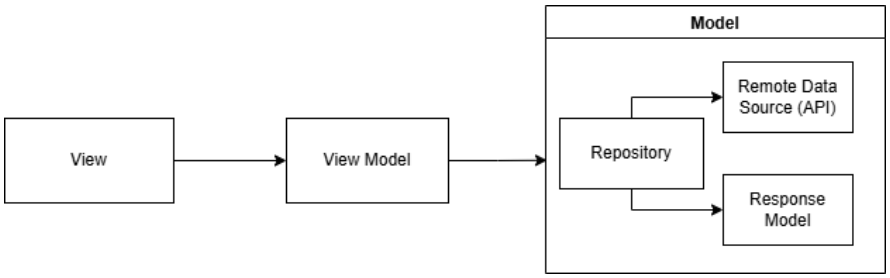
Pengembangan aplikasi *GlucoFit* menggunakan pendekatan arsitektur Model-View-ViewModel (MVVM) untuk memisahkan tanggung jawab antar komponen dalam aplikasi, sehingga kode lebih terstruktur, mudah dirawat, dan dapat dikembangkan lebih lanjut. Dalam penerapannya, arsitektur ini terdiri dari tiga lapisan utama, yaitu View, ViewModel, dan Model.

Lapisan **View** terdiri dari komponen-komponen antarmuka pengguna seperti `Activity` atau `Fragment` yang bertugas menampilkan informasi dan menerima interaksi dari pengguna. View akan berkomunikasi dengan ViewModel**,** yang bertindak sebagai penghubung antara View dan Model. ViewModel menyimpan dan mengelola data yang dibutuhkan oleh View, serta memanggil fungsi-fungsi dari Repository untuk mendapatkan atau mengirimkan data.

Selanjutnya, bagian Model mencakup beberapa komponen penting, yaitu Repository, Remote Data Source (API)**,** dan Response Model. Repository bertanggung jawab sebagai pengelola data yang menjembatani komunikasi antara ViewModel dan sumber data eksternal. Dalam implementasinya, Repository menggunakan Retrofit, yaitu library HTTP client berbasis REST yang populer di Android, untuk melakukan komunikasi jaringan secara efisien. Retrofit

memungkinkan pengiriman dan pengambilan data dari API yang dibangun menggunakan Express.js dan di deploy melalui Google Cloud Run. Melalui konfigurasi `ApiService`, setiap endpoint API dapat dipanggil menggunakan anotasi Retrofit seperti `@GET`, `@POST`, atau `@PUT`, dan hasilnya secara otomatis diubah menjadi objek Kotlin melalui proses deserialisasi.

Data yang diterima dari API kemudian akan dikonversi ke dalam Response Model, yaitu struktur data berupa data class yang digunakan untuk merepresentasikan isi dari response API agar dapat diproses lebih lanjut oleh aplikasi. Alur komunikasi antara komponen-komponen tersebut divisualisasikan dalam Gambar 3.5, yang menunjukkan bagaimana data mengalir dari View ke ViewModel, diteruskan ke Repository, dan akhirnya ke Remote Data Source dan Response Model yang berada pada lapisan Model.




Gambar 3. 5 Arsitektur Perangkat Lunak

### 3.2.5 Tahap Pengujian (Test)

Setelah setiap sprint selesai, dilakukan pengujian terhadap aplikasi untuk memastikan kualitasnya. Pengujian yang dilakukan menggunakan

A. **Blackbox *Testing***

Blackbox Testing merupakan suatu metode pengujian yang berfokus pada spesifikasi fungsional dari suatu perangkat lunak. Proses Blackbox Testing bertujuan untuk memastikan bahwa setiap fitur berfungsi dengan baik serta memenuhi kebutuhan pengguna tanpa melakukan pemeriksaan terhadap kode program. Pengujian ini dilakukan secara berulang pada setiap siklus iterasi, sehingga masalah dapat terdeteksi dan diperbaiki sejak dini (Aldisa & Maulana, 2022).

**B. System Usability Scale (SUS)**

System Usability Scale (SUS) adalah metode pengujian yang digunakan untuk mengukur tingkat kemudahan penggunaan (usability) suatu aplikasi dari perspektif pengguna. Pengujian ini dilakukan dengan menggunakan kuesioner yang disebarkan melalui Google Form. Kuesioner tersebut terdiri dari 10 pertanyaan yang harus dijawab oleh pengguna setelah mereka mengoperasikan aplikasi "Pemantau Konsumsi Gula Harian". Daftar pertanyaan untuk pengujian dengan metode SUS dapat dilihat pada Tabel 3.2.

Tabel 3. 2 Pertanyaan Kuesioner Pengujian System Usability Scale (SUS)

| Kode | Pertanyaan |
|---|---|
| K1 | Saya merasa aplikasi ini mudah digunakan |
| K2 | Saya menemukan aplikasi ini terlalu rumit. |
| K3 | Saya merasa aplikasi ini mudah dioperasikan. |
| K4 | Saya membutuhkan bantuan teknis untuk menggunakan aplikasi ini. |
| K5 | Saya menemukan berbagai fitur dalam aplikasi ini terintegrasi dengan baik. |
| K6 | Saya merasa ada terlalu banyak ketidak konsistenan dalam aplikasi ini. |
| K7 | Saya dapat membayangkan bahwa kebanyakan orang akan belajar menggunakan aplikasi ini dengan cepat. |
| K8 | Saya merasa aplikasi ini tidak nyaman digunakan. |
| K9 | Saya merasa sangat percaya diri saat menggunakan aplikasi ini. |
| K10 | Saya perlu mempelajari banyak hal sebelum dapat menggunakan aplikasi ini dengan efektif. |

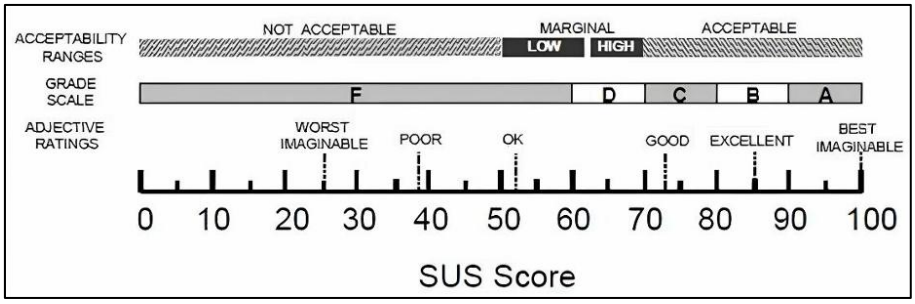
Setiap pertanyaan dari kuesioner akan dijawab menggunakan skala Likert dengan rentang nilai 1 sampai 5. Skor dari skala Likert yang digunakan dapat dilihat pada Tabel 3.3.

Tabel 3. 3 Tabel Skor Penilaian Skala Likert

| No | Kriteria | Skor |
|---|---|---|
| 1. | Sangat Tidak Setuju (STS) | 1 |
| 2. | Tidak Setuju (TS) | 2 |
| 3. | Ragu-ragu (RG) | 3 |
| 4. | Setuju (ST) | 4 |
| 5. | Sangat Setuju (SS) | 5 |

Skor dari pertanyaan pengujian pada Tabel 3.2 kemudian akan diubah dan dihitung sesuai dengan rumus perhitungan SUS. Rumus perhitungan SUS ditunjukkan pada persamaan (3.1) (Brooke, 1995)

$$skor = ((K1 - 1) + (5 - K2) + (K3 - 1) + (5 - K4) + (K5 - 1) + (5 - K6) + (K7 - 1) + (5 - K8) + (K9 - 1) + (5 - K10)).2{,}5 \quad (3.1)$$




Gambar 3. 6 Skala Skor Pengujian SUS

Gambar 3.4 menunjukkan bahwa skor SUS memiliki rentang nilai antara 0 hingga 100, yang dikategorikan ke dalam tiga aspek utama: tingkat penerimaan, skala nilai huruf, dan deskripsi penilaian. Tingkat penerimaan dibagi menjadi tiga kategori, yaitu “Not Acceptable” untuk skor di bawah 50, “Marginal” untuk skor 50-70, dan “Acceptable” untuk skor di atas 70. Selain itu, skor SUS juga dikonversi ke dalam skala huruf dari F hingga A, di mana nilai yang lebih tinggi menunjukkan tingkat kegunaan yang lebih baik. Untuk deskripsi penilaian, skor di bawah 25 dikategorikan sebagai “Worst Imaginable”, skor 25-39 sebagai “Poor”, skor 40-59 sebagai “OK”, skor 60-79 sebagai “Good”, skor 80-89 sebagai “Excellent”, dan skor di atas 90 sebagai “Best Imaginable”. Dengan klasifikasi ini, skor SUS memberikan pemahaman yang lebih jelas mengenai tingkat kegunaan suatu sistem atau aplikasi.

# BAB IV
# HASIL DAN PEMBAHSAN

Pada bab ini, akan dipaparkan hasil yang diperoleh dari pengembangan sistem berbasis Android untuk memantau konsumsi gula harian, yang bertujuan untuk mencegah resiko diabetes. Hasil tersebut dicapai melalui beberapa tahapan pengembangan menggunakan metodologi Agile, yang meliputi analisis kebutuhan pengguna, perencanaan fitur dan pengembangan , pembangunan aplikasi, serta pengujian.

Bab ini akan menjelaskan hasil yang diperoleh pada setiap tahapan, mulai dari identifikasi kebutuhan pengguna hingga evaluasi sistem yang telah dikembangkan. Penulis akan menguraikan bagaimana pendekatan Agile yang bersifat iteratif dan fleksibel telah diterapkan dalam pengembangan aplikasi ini.

## 4.1 Tahap Survei

Berdasarkan hasil survei yang dilakukan terhadap 58 responden (remaja dan dewasa muda), ditemukan beberapa kebutuhan utama pengguna terkait aplikasi pemantau konsumsi gula harian. Hasil ini memberikan gambaran yang sangat berharga mengenai preferensi pengguna, baik dari segi fitur maupun tujuan penggunaan aplikasi. Sebagian besar responden memiliki latar belakang yang berbeda, dengan banyak di antaranya yang memiliki kesadaran tinggi terhadap pentingnya pemantauan konsumsi gula untuk alasan kesehatan, meskipun hanya sedikit yang memiliki kondisi kesehatan tertentu yang memerlukan pemantauan gula secara rutin.

Berikut adalah tabel yang merangkum hasil survei terkait kebutuhan pengguna dalam pengembangan aplikasi ini:

Tabel 4.1 Hasil Survai

| Pertanyaan | Hasil Jawaban Responden |
|---|---|
| Apakah Anda pernah menggunakan aplikasi untuk memantau kesehatan Anda sebelumnya? | Sebagian besar responden (48.3%) mengatakan tidak pernah menggunakan aplikasi untuk memantau kesehatan, sementara 39.7% mengatakan mereka menggunakan aplikasi kesehatan kadang-kadang, dan 12.1% mengatakan mereka sering menggunakannya. |
| Apakah Anda memiliki kondisi kesehatan tertentu (misalnya, | Sebagian besar responden (94.8%) tidak memiliki kondisi kesehatan khusus yang memerlukan pemantauan, sementara hanya 5.2% yang mengatakan |

| | |
|---|---|
| diabetes) yang memerlukan pemantauan gula? | mereka memiliki kondisi seperti diabetes yang membutuhkan pemantauan gula harian. |
| Apa tujuan Anda untuk memantau konsumsi gula harian Anda? | Mayoritas responden (53.4%) mengatakan bahwa tujuan mereka adalah untuk lebih sadar akan pola makan mereka, sementara 24.1% beralasan untuk kesehatan (misalnya, mencegah atau mengelola diabetes). Sisanya, 13.8% ingin meningkatkan energi dan kebugaran. |
| Seberapa sering Anda memperhatikan konsumsi gula harian Anda? | Sebagian besar responden (43.1%) memerhatikan konsumsi gula mereka setiap hari karena alasan kesehatan, sedangkan 29.3% hanya memperhatikannya beberapa kali seminggu. Hanya 1.7% yang tidak pernah memeprhatikan konsumsi gula mereka. |
| Apakah Anda lebih suka mencatat konsumsi gula harian secara manual atau mengandalkan fitur otomatis dari aplikasi? | Lebih dari setengah responden (53.4%) menginginkan pencatatan gula yang dapat dilakukan otomatis, sementara 22.4% lebih suka mencatatnya secara manual, dan 24.1% ingin kedua opsi tersebut tersedia. |
| Seberapa penting bagi Anda untuk mendapatkan notifikasi ketika konsumsi gula Anda hampir mencapai batas harian? | 82.8% responden menyatakan bahwa pemberian notifikasi sangat membantu dan penting, sementara sisanya 17.2% merasa notifikasi mungkin berguna namun tidak terlalu penting. |
| Apakah Anda memiliki target konsumsi gula harian tertentu? | Sebanyak 37.9% responden memiliki target konsumsi gula harian dan ingin aplikasi ini membantu mencapainya, sementara 41.4% tidak memiliki target khusus dan 20.7% hanya ingin mencatat konsumsi gula mereka tanpa target tertentu. |
| Apakah fitur yang memungkinkan Anda melihat riwayat konsumsi gula harian menarik bagi Anda? | Sebagian besar responden (93.2%) tertarik dengan fitur yang memungkinkan mereka melihat riwayat konsumsi gula harian, dengan 46.6% mengatakan fitur |

| | |
|---|---|
| | ini sangat menarik dan 46.6% lainnya menganggapnya cukup menarik. |
| Bagaimana biasanya Anda mengetahui jumlah gula yang Anda konsumsi setiap hari? | Sebanyak 39.7% responden mengetahui jumlah gula mereka melalui informasi pada label produk, 34.5% menggunakan perkiraan pribadi, dan 34.5% lainnya menginginkan aplikasi untuk menyediakan fitur pemindai kandungan gula pada produk. |
| Apabila akan dibangun aplikasi baru, apakah Anda ingin aplikasi yang dapat memberi saran atau peringatan saat konsumsi gula Anda melampaui batas aman? | Sebagian besar responden (82.8%) menginginkan aplikasi yang memberikan saran atau peringatan saat konsumsi gula mereka melampaui batas aman, dengan 17.2% lainnya merasa fitur ini mungkin berguna. |
| Apakah menurut Anda aplikasi ini perlu menyediakan artikel dan tips terkait pola makan sehat serta pengelolaan gula? | Sebagian besar responden (72.4%) merasa bahwa aplikasi ini perlu menyediakan artikel dan tips terkait pola makan sehat serta pengelolaan gula, sementara 25.9% menganggap fitur ini sangat diperlukan dan 1.7% mengatakan tidak terlalu diperlukan. |

Berdasarkan hasil survei ini, kesimpulan yang dapat diambil adalah bahwa meskipun sebagian besar responden belum terbiasa menggunakan aplikasi pemantau kesehatan, mereka menunjukkan minat yang tinggi terhadap fitur-fitur aplikasi yang dapat membantu mereka lebih sadar tentang konsumsi gula harian. Sebagian besar responden menginginkan aplikasi yang menyediakan pencatatan otomatis untuk konsumsi gula, notifikasi terkait batas konsumsi gula harian, serta fitur yang memungkinkan mereka melihat riwayat konsumsi gula. Selain itu, sebagian besar juga tertarik pada aplikasi yang dapat memberikan saran atau peringatan ketika konsumsi gula melampaui batas yang aman.

Fitur-fitur edukatif, seperti artikel dan tips terkait pola makan sehat dan pengelolaan gula, juga dianggap penting oleh mayoritas responden. Hal ini menunjukkan bahwa ada kebutuhan besar untuk aplikasi yang tidak hanya fokus pada pemantauan konsumsi gula, tetapi juga memberikan informasi yang dapat membantu pengguna menjaga kesehatan mereka secara lebih menyeluruh. Dengan demikian, pengembangan aplikasi pemantau konsumsi gula harian yang memenuhi kebutuhan ini berpotensi untuk memperoleh penerimaan yang baik di pasar.

## 4.2 Analisis Kebutuhan

Berdasarkan hasil analisis kebutuhan pengguna yang didapatkan melalui survei terhadap 58 responden, dapat disimpulkan bahwa fitur-fitur yang diinginkan oleh pengguna untuk aplikasi pemantauan konsumsi gula harian dapat dibagi menjadi dua kategori: fitur prioritas (utama) dan fitur penunjang (pendukung).

### 4.2.1 Fitur Prioritas

Fitur-fitur berikut merupakan kebutuhan utama yang harus dipenuhi dalam pengembangan aplikasi ini karena langsung berhubungan dengan tujuan aplikasi untuk memantau konsumsi gula harian dan meningkatkan kesehatan pengguna:

1. Informasi Pengguna untuk Menghitung Kalori dan Gula Harian

   Aplikasi akan mengumpulkan informasi terkait pola makan pengguna untuk menghitung asupan kalori dan gula harian, yang akan digunakan sebagai dasar untuk memantau kesehatan secara lebih akurat.

2. Pencatatan Konsumsi Gula Harian (Otomatis dan Manual)

   Pencatatan konsumsi gula harian dapat dilakukan secara otomatis, di mana pengguna cukup mengunggah foto makanan dan aplikasi akan menganalisis kandungan gula dan kalori dalam makanan tersebut. Untuk pencatatan manual, pengguna dapat langsung memasukkan data makanan yang mereka konsumsi, memungkinkan pengelolaan yang lebih fleksibel.

3. Notifikasi Konsumsi Gula

   Aplikasi akan memberikan notifikasi kepada pengguna apabila konsumsi gula mereka mendekati atau melampaui batas yang disarankan. Selain itu, pengguna juga akan diingatkan untuk melakukan pencatatan konsumsi gula secara rutin sebanyak tiga kali sehari, dengan pengingat pada jam 08:00, 12:00, dan 18:00.

4. Edukasi

   Aplikasi ini menyediakan artikel dan tips terkait pola makan sehat dan pengelolaan gula yang efektif, guna memberikan pengetahuan lebih kepada pengguna dalam menjaga gaya hidup yang lebih sehat.

5. Monitoring Konsumsi Gula Harian

   Fitur ini memungkinkan pengguna untuk memantau statistik konsumsi gula harian mereka. Selain itu, riwayat konsumsi gula akan disajikan dengan ikon status yang menggambarkan perasaan pengguna (misalnya: senang, marah, biasa), yang bertujuan

untuk meningkatkan kesadaran dan kepedulian pengguna terhadap pentingnya pengelolaan konsumsi gula yang sehat.

6. Chatbot untuk Membantu Pengguna dengan Pertanyaan Seputar Kesehatan

   Fitur chatbot akan membantu pengguna dengan menjawab pertanyaan seputar pola makan sehat, pengelolaan gula, dan topik kesehatan lainnya, memberikan kenyamanan dan kemudahan dalam memperoleh informasi terkait kesehatan.

7. Panduan Konsumsi Gula dan Status Pengguna

   Aplikasi akan memberikan rekomendasi konsumsi gula harian yang aman, berdasarkan standar yang ditetapkan oleh WHO dan Kementerian Kesehatan Republik Indonesia. Selain itu, aplikasi juga memberikan penjelasan tentang tingkat emotikon (misalnya senang, marah, biasa) yang terkait dengan konsumsi gula pengguna, untuk membantu mereka lebih memahami dampak konsumsi gula terhadap kesehatan mereka.

8. Scan Bar*Code*

   Aplikasi memungkinkan pengguna untuk memindai label makanan atau minuman menggunakan kamera ponsel mereka, guna mendeteksi dan menampilkan informasi kandungan gula secara instan.

### **4.2.2** Fitur Penunjang (Pendukung)

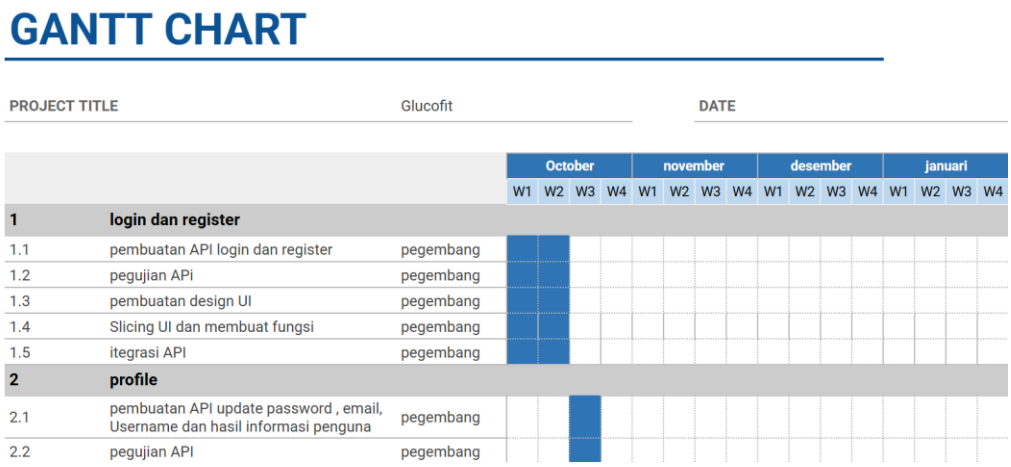
Selain fitur utama di atas, terdapat beberapa fitur penunjang yang mendukung pengalaman pengguna dan mempermudah akses ke aplikasi. Meskipun fitur-fitur ini tidak berhubungan langsung dengan pemantauan konsumsi gula, mereka tetap penting untuk memastikan aplikasi berjalan dengan lancar dan memberikan kenyamanan bagi pengguna:

1. Halaman login
2. Halaman *onboarding*
3. Halaman register
4. Halaman profile

Fitur prioritas yang telah disebutkan di atas harus menjadi fokus utama dalam pengembangan aplikasi karena mereka berkaitan langsung dengan tujuan aplikasi, yaitu memantau konsumsi gula dan meningkatkan kesehatan pengguna. Di sisi lain, fitur pendukung seperti halaman onboarding, login, dan registrasi sangat penting untuk memberikan pengalaman pengguna yang mulus dan aman, meskipun mereka lebih berfokus pada aksesibilitas dan pengalaman awal pengguna daripada pada fungsi utama aplikasi.

### 4.3 Tahap perancangan (plan)

Pada tahap ini akan dilakukan penjabaran secara jelas terkait perjalanan pengembangan proyek aplikasi pemantauan konsumsi gula harian. Di dalam pengembangan proyek aplikasi ini, rencana-rencana yang akan dilakukan selama pengembangan diuraikan menggunakan grafik Gantt Chart yang dapat dilihat pada Gambar 4.1 Gantt Chart memudahkan dalam melakukan monitoring proyek yang sedang dikerjakan. Gantt Chart ini akan di-*update* setiap ada progress atau pengerjaan yang telah selesai.




Gambar 4. 1 Gantt Chart

Berdasarkan Gantt Chart, pengembangan aplikasi dibagi menjadi beberapa sprint. Setiap sprint berfokus pada pengembangan fitur tertentu dalam periode waktu yang telah ditentukan. Berikut adalah rincian fitur yang akan dikembangkan dalam setiap sprint seperti pada tabel

Tabel 4. 2 Sprint

| **Sprint** | **Fitur yang Dikembangkan** | **Deskripsi** |
|---|---|---|
| Sprint 1 | Login dan Register | Fitur untuk memungkinkan pengguna melakukan login dan pendaftaran di aplikasi. |
| | Informasi Pengguna untuk Menghitung Kalori dan Gula Harian | Aplikasi akan mengumpulkan informasi terkait pola makan pengguna untuk menghitung asupan kalori dan gula harian, sebagai dasar untuk memantau kesehatan secara lebih akurat. |
| Sprint 2 | Pencatatan Konsumsi Gula Harian (Otomatis/Manual) | Pengembangan fitur pencatatan konsumsi gula harian, baik secara otomatis (melalui foto makanan yang dianalisis untuk kandungan gula |

| | | |
|---|---|---|
| | | dan kalori) maupun manual (pengguna memasukkan data makanan yang dikonsumsi). |
| | Notifikasi Konsumsi Gula | Fitur yang memberikan notifikasi kepada pengguna apabila konsumsi gula mendekati atau melampaui batas yang disarankan, serta pengingat untuk pencatatan gula pada jam 08:00, 12:00, dan 18:00. |
| Sprint 3 | Edukasi | Aplikasi menyediakan artikel dan tips terkait pola makan sehat dan pengelolaan gula yang efektif, untuk memberikan pengetahuan lebih kepada pengguna dalam menjaga gaya hidup yang lebih sehat. |
| | Monitoring Konsumsi Gula Harian | Fitur yang memungkinkan pengguna memantau statistik konsumsi gula harian mereka. Riwayat konsumsi gula juga akan disajikan dengan ikon status untuk menggambarkan perasaan pengguna, seperti senang, marah, atau biasa. |
| Sprint 4 | Chatbot untuk Membantu Pengguna dengan Pertanyaan Seputar Kesehatan | Chatbot ini membantu menjawab pertanyaan pengguna terkait pola makan sehat, pengelolaan gula, dan topik kesehatan lainnya, untuk memberikan kenyamanan dan kemudahan dalam mendapatkan informasi. |
| Sprint 5 | Panduan Konsumsi Gula dan Status Pengguna | Fitur ini memberikan rekomendasi konsumsi gula harian yang aman sesuai dengan standar WHO dan Kementerian Kesehatan RI, serta memberikan penjelasan tentang tingkat emoticon (misalnya senang, marah, biasa) yang terkait dengan konsumsi gula pengguna. |
| Sprint 6 | Artikel dan Tips Pola Makan Sehat | Penyediaan artikel dan tips mengenai pola makan sehat, serta cara mengelola konsumsi gula yang |

| | | |
|---|---|---|
| | | efektif, untuk mendukung edukasi pengguna dalam menjaga kesehatan secara keseluruhan. |

## 4.4 Tahap Rancangan (*Design*)

### 4.4.1 *Use Case Diagram*

*Use Case Diagram* dibuat sebagai gambaran interaksi pengguna dengan sistem. *Use Case Diagram* dari aplikasi yang akan dibangun ditunjukkan pada Gambar 4.2.




Gambar 4. 2 *Use Case Diagram*

Penjelasan mengenai actor yang ada pada use case dapat di lihat pada Tabel 4.3

Tabel 4. 3 Aktor pada *Use Case Diagram*

| Aktor | Deskripsi |
|---|---|
| Pengguna | individu yang menggunakan aplikasi untuk mengelola dan memantau kesehatan mereka terkait konsumsi gula |

Penjelasan mengenai fungsionalitas yang terdapat pada *Use Case Diagram* dapat ditemukan pada Tabel 4.4

Tabel 4. 4 Fungsionalitas Fitur

| **No** | ***Use Case*** | **Keterangan** |
|---|---|---|
| UC1 | Login / Register | Aksi untuk masuk ke dalam aplikasi atau membuat akun baru jika belum memiliki akun. |
| UC2 | Pencatatan Konsumsi Gula Harian | Aksi untuk mencatat konsumsi gula harian, baik secara otomatis dengan mengunggah foto atau manual dengan memasukkan data makanan yang dikonsumsi. |
| UC3 | Menambah Data Gula Otomatis | Aksi untuk menambahkan data konsumsi gula secara otomatis menggunakan foto makanan yang diunggah oleh pengguna. |
| UC4 | Menambah Data Gula Manual | Aksi untuk menambahkan data konsumsi gula secara manual, dengan memasukkan informasi makanan yang dikonsumsi oleh pengguna. |
| UC5 | Monitoring Konsumsi Gula Harian | Aksi untuk memantau konsumsi gula harian, termasuk melihat riwayat konsumsi gula dan statistik konsumsi gula pengguna. |
| UC6 | Melihat Riwayat Konsumsi | Aksi untuk melihat daftar konsumsi gula yang telah dicatat sebelumnya. |
| UC7 | Melihat Statistik | Aksi untuk melihat statistik terkait konsumsi gula pengguna dari waktu ke waktu. |
| UC8 | Menggunakan Chatbot | Aksi untuk berinteraksi dengan chatbot guna mendapatkan informasi terkait pola makan sehat dan konsumsi gula. |
| UC9 | Melihat Edukasi | Aksi untuk melihat artikel dan tips terkait pola makan sehat dan pengelolaan konsumsi gula. |
| UC10 | Informasi Pengguna untuk Menghitung Kalori dan Gula Harian | Aksi unt uk melihat informasi mengenai jumlah kalori dan gula yang dikonsumsi berdasarkan data yang dimasukkan oleh pengguna. |

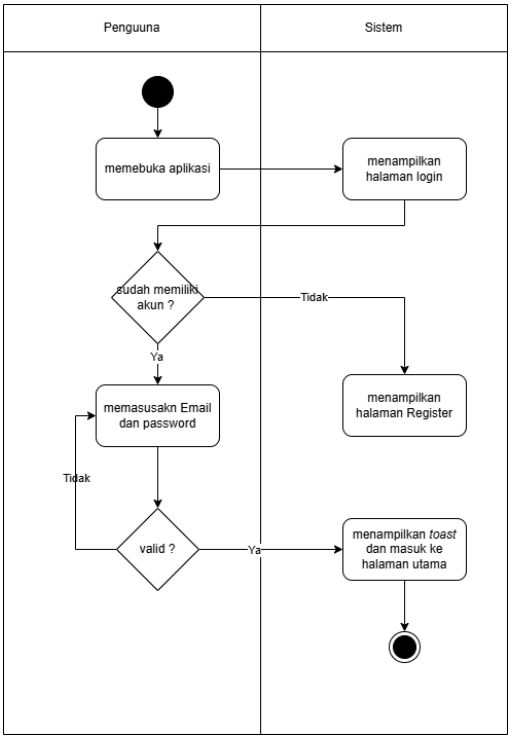
| | | |
|---|---|---|
| UC11 | Menerima Notifikasi | Aksi untuk menerima notifikasi yang mengingatkan pengguna untuk mencatat konsumsi gula mereka pada waktu yang telah ditentukan dan ketika konsumsi gula pengguna mendekati atau melebihi batas yang disarankan |
| UC12 | Melihat Account | Aksi untuk melihat informasi profil pengguna yang mencakup nama pengguna, alamat email, dan informasi lainnya. |
| UC13 | Melihat Hasil Informasi Pengguna | Aksi untuk melihat hasil dari informasi yang telah dimasukkan oleh pengguna, seperti riwayat konsumsi gula dan kalori. |
| UC14 | Mengubah Username | Aksi untuk mengubah nama pengguna dalam aplikasi sesuai dengan preferensi pengguna. |
| UC15 | Mengubah Email | Aksi untuk mengubah alamat email yang terdaftar pada akun pengguna. |
| UC16 | Mengubah Password | Aksi untuk mengganti kata sandi pengguna untuk keamanan aplikasi. |
| UC17 | Scan Bar*Code* | Aksi untuk melihatkan hasil informasi label nutrisi pada produk. |

### 4.4.2 *Activity Diagram*

Berikut *Activity Diagram* pada sistem ini menggambarkan proses interaksi antara sistem dan pengguna yang terkait dengan fitur-fitur yang tersedia dalam sistem tersebut.

f. *Activity Diagram* Login

*Activity Diagram* login menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam proses login ke dalam sistem. Pada diagram ini, pengguna akan diminta untuk memasukkan email dan password, serta sistem akan memverifikasi kevalidan data yang dimasukkan. Diagram ini dapat dilihat pada Gambar 4.3.

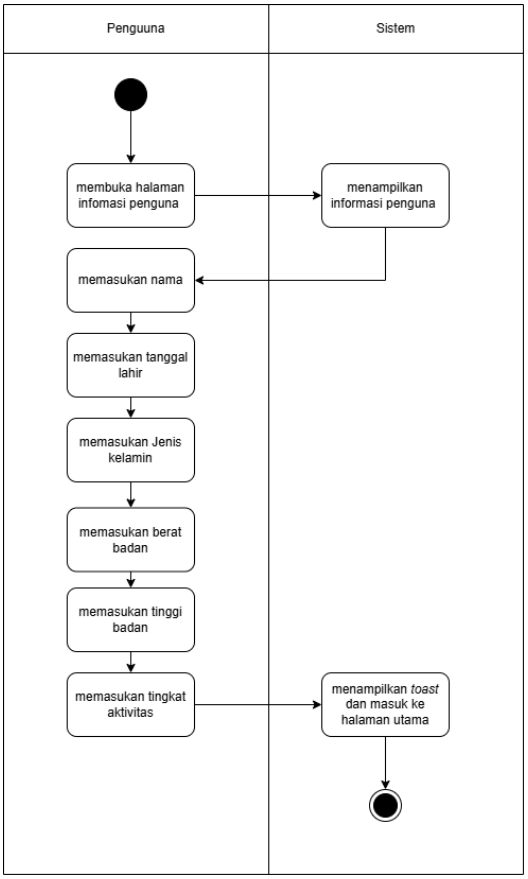



Gambar 4. 3 *Activity Diagram* Login

Pada Gambar 4.3, dapat dilihat bahwa proses dimulai ketika pengguna membuka aplikasi, yang kemudian menampilkan halaman login. Setelah halaman login muncul, sistem akan mengecek apakah pengguna sudah memiliki akun atau belum. Jika pengguna sudah memiliki akun, mereka akan diminta untuk memasukkan email dan password. Setelah data dimasukkan, sistem akan melakukan pengecekan terhadap kevalidan email dan password tersebut. Apabila data yang dimasukkan valid, sistem akan menampilkan notifikasi berupa toast dan mengarahkan pengguna ke halaman utama. Namun, jika data yang dimasukkan tidak valid, pengguna akan diminta untuk mengulang proses memasukkan email dan password. Jika pengguna belum memiliki akun, sistem akan mengarahkan mereka ke halaman registrasi untuk membuat akun baru.

g. *Activity Diagram* Informasi Pengguna untuk Menghitung Kalori dan Gula Harian

*Activity Diagram* informasi pengguna menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam mengisi informasi pribadi yang diperlukan dalam sistem. Pada Gambar 4.4, dijelaskan langkah-langkah yang dilakukan oleh pengguna untuk memasukkan informasi pribadi mereka ke dalam sistem.

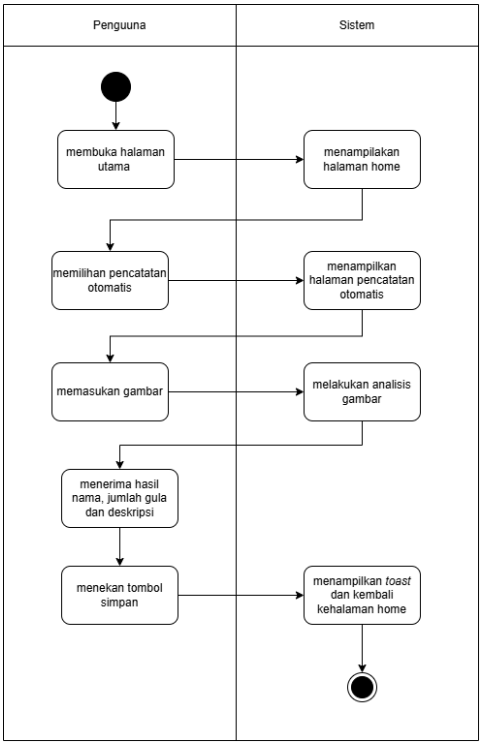



Gambar 4. 4 *Activity Diagram* Informasi Pengguna

Pada Gambar 4.4, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman informasi pengguna, yang kemudian sistem menampilkan halaman Informasi pengguna. Setelah itu, pengguna diminta untuk memasukkan berbagai data pribadi, dimulai dengan nama, diikuti dengan tanggal lahir, jenis kelamin, berat badan, tinggi badan, dan terakhir tingkat aktivitas pengguna. Setelah semua informasi dimasukkan, sistem akan menampilkan notifikasi berupa toast yang menandakan bahwa proses berhasil, dan kemudian mengarahkan pengguna ke halaman utama. Setiap langkah di diagram ini menggambarkan alur pengisian data yang harus dilakukan oleh pengguna secara berurutan.

h. *Activity Diagram* Menambah Data Gula Otomatis

*Activity Diagram* pencatatan otomatis menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk memasukkan gambar dan menerima hasil analisis dari sistem. Pada Gambar 4.5, dijelaskan proses yang dilakukan pengguna untuk mendapatkan hasil pencatatan otomatis

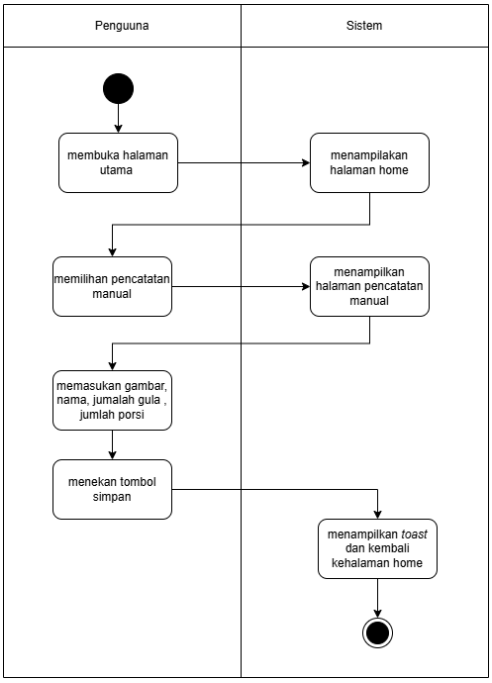



Gambar 4. 5 *Activity Diagram* Data Gula Otomatis

Pada Gambar 4.5, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman utama aplikasi, yang kemudian sistem menampilkan halaman home. Setelah itu, pengguna memilih opsi pencatatan otomatis, yang akan menampilkan halaman pencatatan otomatis. Pengguna kemudian memasukkan gambar yang akan dianalisis oleh sistem. Setelah gambar dimasukkan, sistem akan melakukan analisis dan menghasilkan informasi berupa nama, jumlah gula, dan deskripsi terkait gambar tersebut. Hasil ini akan diterima oleh pengguna, yang kemudian menekan tombol simpan untuk menyimpan data yang telah dianalisis. Setelah itu, sistem akan menampilkan notifikasi berupa toast sebagai tanda bahwa data berhasil disimpan, dan mengarahkan pengguna kembali ke halaman utama.

i. *Activity Diagram* Menambah Data Gula manual

*Activity Diagram* pencatatan manual menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam memasukkan data secara manual ke dalam sistem. Pada Gambar 4.6, dijelaskan proses yang dilakukan pengguna untuk melakukan pencatatan manual.

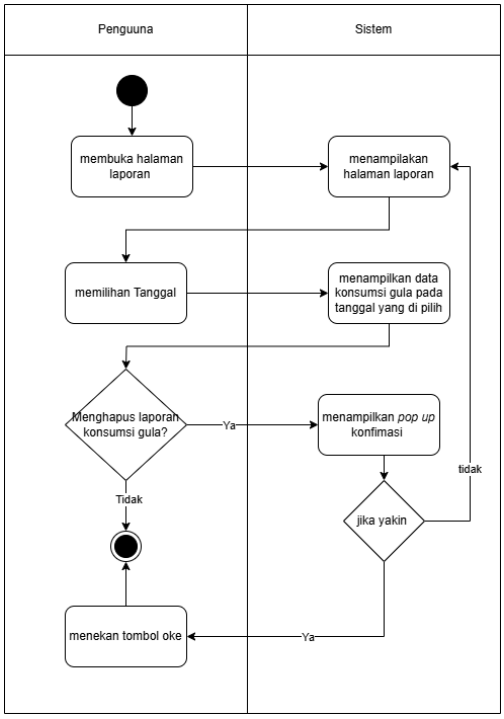



Gambar 4. 6 *Activity Diagram* Gula Manual

Pada Gambar 4.6, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman utama aplikasi, yang kemudian system menampilkan halaman home. Setelah itu, pengguna memilih opsi pencatatan manual, yang akan menampilkan halaman pencatatan manual. Pengguna kemudian memasukkan gambar, nama, jumlah gula, dan jumlah porsi yang terkait dengan data yang ingin dicatat. Setelah semua data dimasukkan, pengguna menekan tombol simpan untuk menyimpan informasi yang telah dimasukkan. Sistem kemudian akan menampilkan notifikasi berupa toast sebagai tanda bahwa data telah berhasil disimpan dan kemudian mengarahkan pengguna kembali ke halaman utama.

j. *Activity Diagram* Melihat Riwayat Konsumsi

*Activity Diagram* laporan konsumsi gula menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam mengakses laporan konsumsi gula dan mengelola data konsumsi gula yang tercatat. Pada Gambar 4.7, dijelaskan proses yang dilakukan pengguna untuk memilih dan menghapus laporan konsumsi gula.

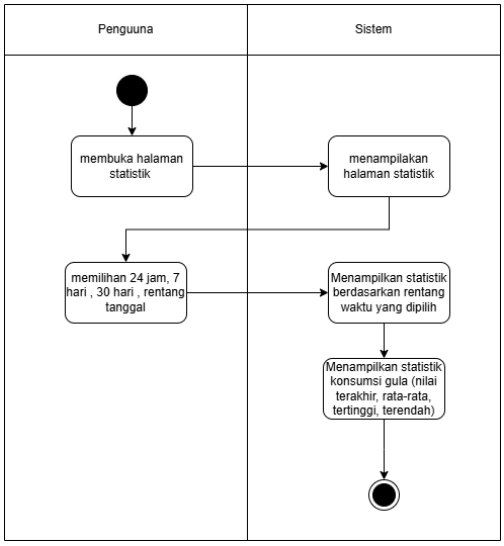



Gambar 4. 7 *Activity Diagram* Melihat Riwayat Konsumsi

Pada Gambar 4.7, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman laporan, yang kemudian system menampilkan halaman laporan. Setelah itu pengunan memilih tanggal yang diinginkan untuk melihat laporan konsumsi gula pada tanggal tersebut. Setelah data laporan pada tanggal yang dipilih ditampilkan, jika ingin melakukan penghapusan sistem menanyakan apakah pengguna ingin menghapus laporan tersebut. Jika pengguna memilih Tidak, mereka bisa langsung melanjutkan tanpa ada perubahan. Jika pengguna memilih Ya, sistem akan menampilkan pop-up konfirmasi untuk memastikan apakah pengguna benar-benar ingin menghapus laporan. Jika pengguna yakin, mereka dapat menekan tombol "OK" untuk menghapus laporan tersebut. Setelah itu, sistem akan mengupdate data dan pengguna dapat melanjutkan ke tindakan berikutnya atau kembali ke halaman utama.

k. *Activity Diagram* Melihat Statistik

*Activity Diagram* statistik konsumsi gula menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk melihat data statistik konsumsi gula berdasarkan rentang waktu tertentu. Pada Gambar 4.8, dijelaskan proses yang dilakukan pengguna untuk mengakses dan menampilkan statistik konsumsi gula.

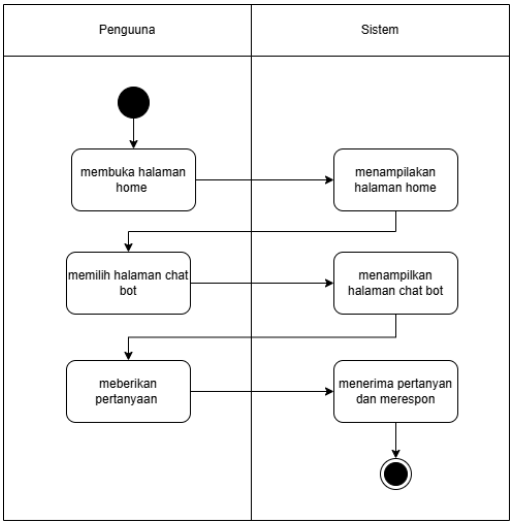



Gambar 4. 8 *Activity Diagram* Melihat Statistik

Pada Gambar 4.8, dapat dilihat bahwa proses dimulai ketika pengguna membuka halaman statistik, yang kemudian sistem menampilkan halaman statistik. Setelah itu, pengguna memilih rentang waktu yang ingin dianalisis, apakah itu 24 jam, 7 hari, 30 hari, atau rentang tanggal tertentu. Berdasarkan pilihan rentang waktu yang dipilih, sistem kemudian menampilkan statistik konsumsi gula, yang mencakup data seperti nilai terakhir, rata-rata, serta nilai tertinggi dan terendah selama periode tersebut.

l. *Activity Diagram* Mengunakan Chat Bot

*Activity Diagram* chat bot menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam berinteraksi dengan sistem melalui chat bot. Pada Gambar 4.9, dijelaskan alur yang terjadi ketika pengguna ingin menggunakan fitur chat bot untuk mendapatkan bantuan.

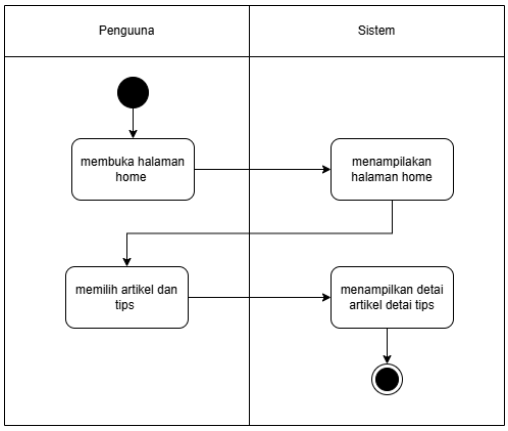



Gambar 4. 9 *Activity Diagram* Menggunakan Chat Bot

Pada Gambar 4.9, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman home, yang kemudian sistem menampilkan halaman home. Setelah itu, pengguna memilih opsi untuk membuka halaman chat bot. Sistem kemudian menampilkan halaman chat bot, di mana pengguna dapat memberikan pertanyaan yang ingin diajukan. Setelah pengguna memberikan pertanyaan, sistem menerima pertanyaan tersebut dan merespons dengan jawaban yang sesuai. Proses ini memungkinkan interaksi langsung antara pengguna dan sistem untuk memberikan informasi atau bantuan yang dibutuhkan.

m. *Activity Diagram* Melihat Edukasi

*Activity Diagram* edukasi menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam mengakses artikel atau tips melalui sistem. Pada Gambar 4.10 dijelaskan alur yang terjadi ketika pengguna memilih artikel atau tips untuk dibaca.



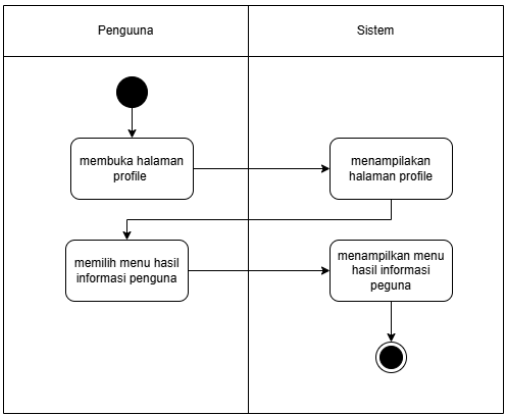

Gambar 4. 10 *Activity Diagram* Melihat Edukasi

Pada Gambar 4.10, dapat dilihat bahwa proses dimulai dengan pengguna membuka halaman home, yang kemudian sistem menampilkan halaman home. Pengguna kemudian

memilih opsi untuk melihat artikel dan tips. Sistem akan menampilkan berbagai artikel dan tips yang tersedia. Pengguna kemudian memilih artikel atau tips yang diinginkan untuk dibaca, dan sistem akan menampilkan detail artikel atau tips tersebut. Dengan demikian, pengguna dapat membaca informasi yang diberikan dalam artikel atau tips tersebut sesuai dengan kebutuhan mereka.

n. *Activity Diagram* Melihat Hasil Informasi Pengguna

*Activity Diagram* menu hasil informasi pengguna menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk mengakses informasi profil mereka. Pada Gambar 4.11, dijelaskan alur yang terjadi ketika pengguna ingin melihat hasil informasi yang terkait dengan profil mereka.

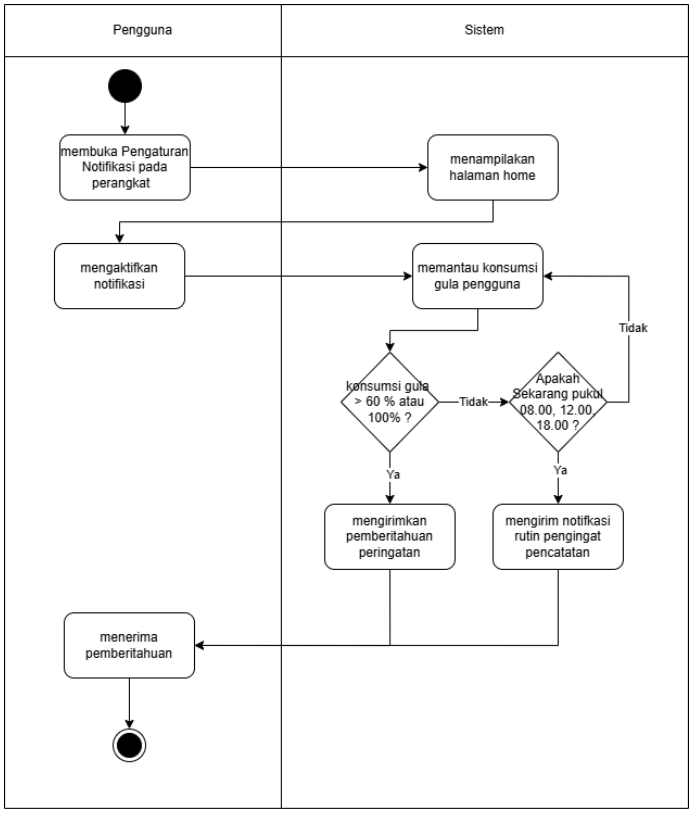



Gambar 4. 11 *Activity Diagram* Melihat Hasil Informasi Penguna

Pada Gambar 4.11, dapat dilihat bahwa proses dimulai ketika pengguna membuka halaman profil, yang kemudian sistem menampilkan halaman profil. Setelah itu, pengguna memilih menu untuk melihat hasil informasi pengguna. Sistem kemudian menampilkan menu yang berisi informasi terkait pengguna, yang bisa mencakup data diri, total kalori dan maksimum jumlah gula konsumsi. Dengan demikian, pengguna dapat melihat detail informasi yang berkaitan dengan profil mereka sesuai dengan pilihan yang telah dipilih.

o. *Activity Diagram* Notifikasi

*Activity Diagram* pengaturan notifikasi menggambarkan langkah-langkah yang dilakukan oleh pengguna dalam mengelola pengaturan notifikasi terkait konsumsi gula. Pada Gambar 4.12, dijelaskan alur yang terjadi ketika pengguna ingin menerima pemberitahuan terkait konsumsi gula.




Gambar 4. 12 *Activity Diagram* Notifikasi

Pada Gambar 4.12, dapat dilihat bahwa proses dimulai ketika pengguna membuka pengaturan notifikasi pada perangkat, yang kemudian menampilkan halaman *home*. Setelah itu, pengguna mengaktifkan notifikasi untuk memantau konsumsi gula mereka.

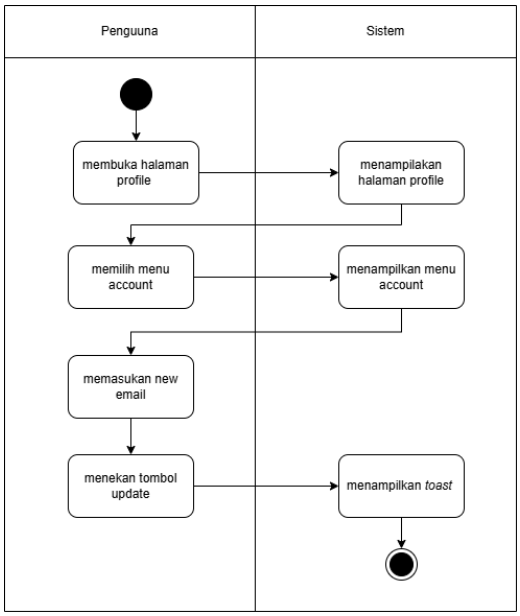
Setelah notifikasi diaktifkan, sistem akan mulai memantau konsumsi gula harian pengguna. Ketika jumlah konsumsi gula mencapai ambang tertentu, yaitu 60% atau 100% dari batas harian, sistem akan memicu pengiriman notifikasi level gula. Secara teknis, pengiriman notifikasi ini menggunakan WorkManager, karena pengingat ini bersifat berbasis kondisi dan membutuhkan proses latar belakang yang andal dan tahan terhadap restart perangkat.

Selain itu, sistem juga menjadwalkan pengiriman notifikasi pada jam tertentu, yaitu pukul 08.00, 12.00, dan 18.00. Untuk notifikasi ini, sistem memanfaatkan **A**larmManager yang digunakan untuk menjadwalkan *time-based notifications*, sehingga pengguna akan mendapatkan pengingat secara rutin pada waktu yang telah ditentukan, tanpa bergantung pada status aplikasi (terbuka atau tertutup).

p. *Activity Diagram* Mengubah Email

*Activity Diagram* pembaruan email pengguna menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk memperbarui alamat email mereka dalam sistem. Pada

Gambar 4.13, dijelaskan alur yang terjadi ketika pengguna ingin memperbarui email yang terdaftar di profil mereka.

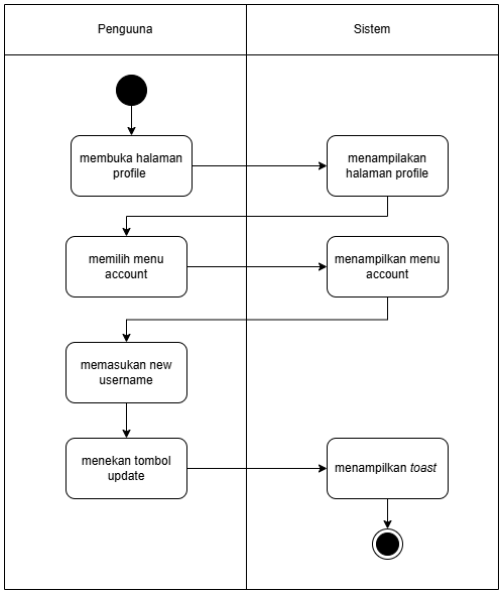



Gambar 4. 13 *Activity Diagram* Mengubah Email

Pada Gambar 4.13, dapat dilihat bahwa proses dimulai ketika pengguna membuka halaman profil, yang kemudian sistem menampilkan halaman profil. Pengguna memilih menu akun untuk mengakses pengaturan akun mereka, dan sistem menampilkan menu akun tersebut. Pengguna kemudian diminta untuk memasukkan alamat email baru. Setelah memasukkan email baru, pengguna menekan tombol *update* untuk memperbarui informasi tersebut. Sistem kemudian menampilkan notifikasi berupa toast sebagai tanda bahwa pembaruan berhasil dilakukan, dan perubahan diterapkan pada akun pengguna.

q. *Activity Diagram* Mengubah Username

*Activity Diagram* pembaruan username pengguna menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk memperbarui username mereka dalam sistem. Pada Gambar 4.14, dijelaskan alur yang terjadi ketika pengguna ingin mengubah username yang terdaftar di profil mereka.

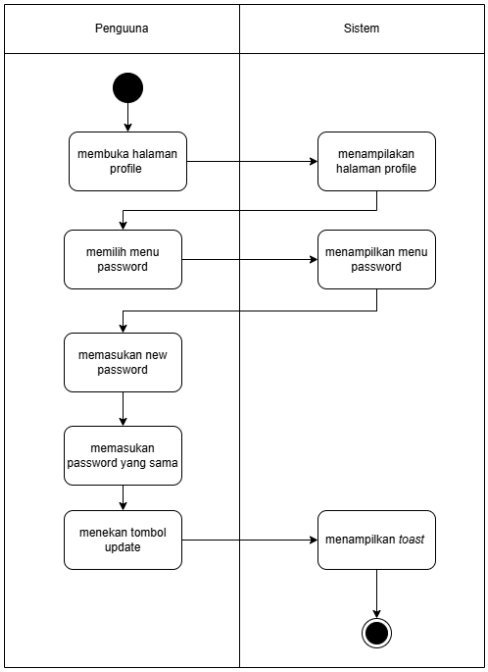



Gambar 4. 14 *Activity Diagram* Mengubah Username

Pada Gambar 4.14, dapat dilihat bahwa proses dimulai ketika pengguna membuka halaman profil, yang kemudian menampilkan halaman profil. Pengguna memilih menu akun untuk mengakses pengaturan akun mereka, dan sistem menampilkan menu akun tersebut. Pengguna kemudian diminta untuk memasukkan username baru (new username). Setelah memasukkan username baru, pengguna menekan tombol *update* untuk memperbarui informasi tersebut. Sistem kemudian menampilkan notifikasi berupa toast sebagai tanda bahwa pembaruan username berhasil dilakukan dan perubahan diterapkan pada akun pengguna.

r. *Activity Diagram* Megubah Password

*Activity Diagram* pembaruan password pengguna menggambarkan langkah-langkah yang dilakukan oleh pengguna untuk memperbarui password mereka dalam sistem. Pada Gambar 4.15, dijelaskan alur yang terjadi ketika pengguna ingin mengganti password yang terdaftar di profil mereka.

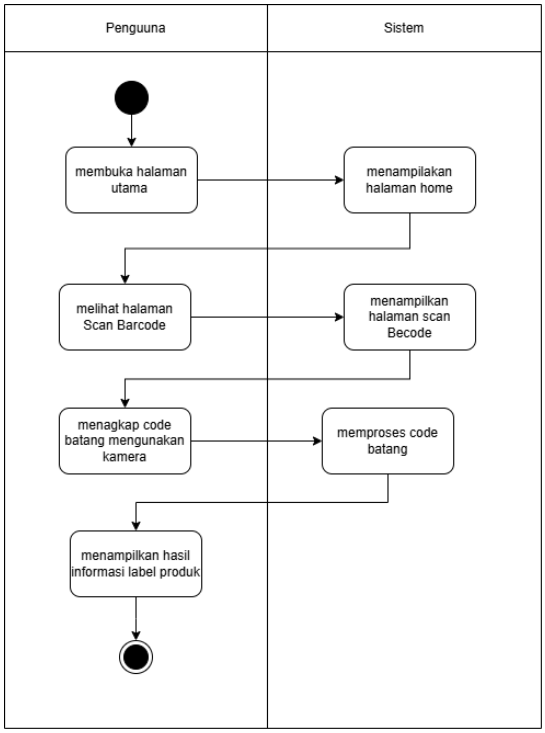



Gambar 4. 15 *Activity Diagram* Mengubah Password

Pada Gambar 4.15, dapat dilihat bahwa proses dimulai ketika pengguna membuka halaman profil, yang kemudian menampilkan halaman profil. Pengguna memilih menu password untuk mengakses pengaturan password mereka, dan sistem menampilkan menu pengaturan password tersebut. Pengguna kemudian diminta untuk memasukkan password baru (new password) dan memastikan untuk memasukkan password yang sama pada kolom konfirmasi. Setelah memasukkan password yang sesuai, pengguna menekan tombol *update* untuk memperbarui password mereka. Sistem kemudian menampilkan notifikasi berupa toast sebagai tanda bahwa pembaruan password berhasil dilakukan dan perubahan diterapkan pada akun pengguna

s. *Activity Diagram* Scan *BarCode*

*Activity Diagram* pemindaian kandungan gula pada produk menggambarkan langkah-langkah yang dilakukan oleh pengguna dan sistem dalam memperoleh informasi kandungan gula dari produk melalui pemindaian kode batang. Pada Gambar 4.16, dijelaskan alur yang terjadi ketika pengguna ingin mengetahui informasi label produk dengan memanfaatkan fitur scan *barCode*.

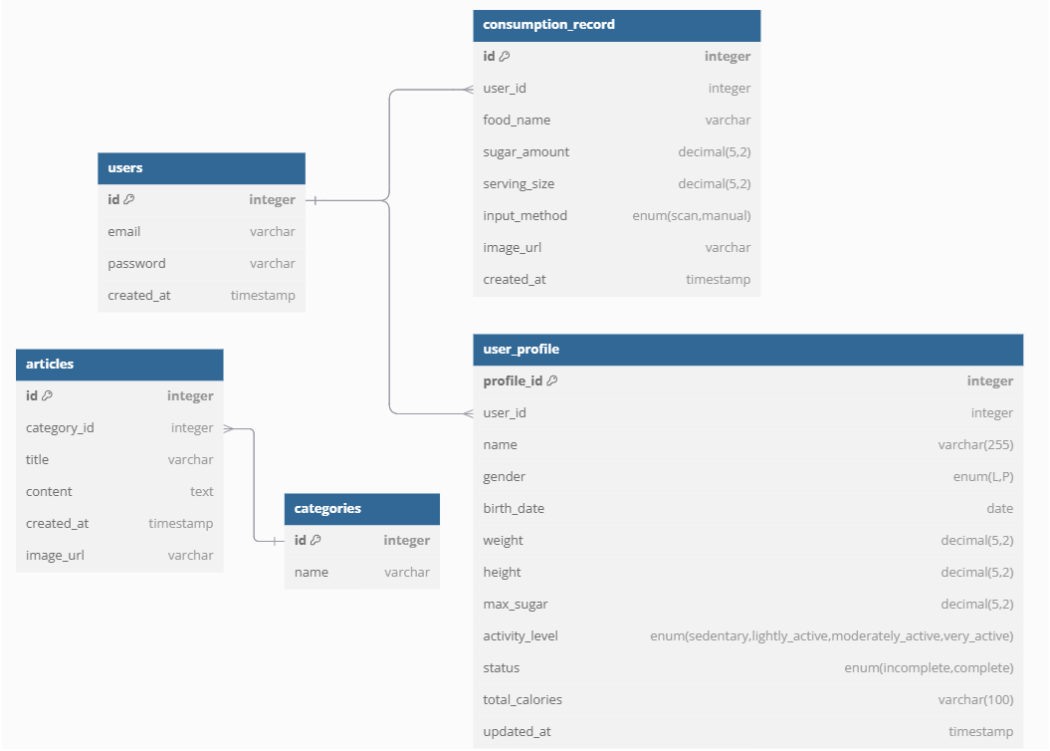



Gambar 4. 16 *Activity Diagram* Scan Bar*Code*

Pada Gambar 4.16, proses dimulai ketika pengguna membuka halaman utama aplikasi. Sistem kemudian merespons dengan menampilkan halaman home. Selanjutnya, pengguna memilih untuk melihat halaman *Scan BarCode*, dan sistem akan menampilkan halaman khusus untuk melakukan pemindaian *barCode*. Setelah halaman pemindaian terbuka, pengguna menangkap kode batang pada produk menggunakan kamera perangkat. Sistem kemudian memproses kode batang tersebut untuk mengidentifikasi produk yang dimaksud. Setelah pemrosesan selesai, sistem menampilkan hasil berupa informasi label produk, termasuk kandungan gula yang terdapat dalam produk tersebut. Proses ini memudahkan pengguna dalam memperoleh informasi gizi secara cepat dan efisien hanya dengan melakukan pemindaian bar*Code*.

### 4.4.3 Rancangan basis Data

Untuk basis data aplikasi, peneliti menggunakan MySQL dengan rancangan tabel yang terdiri dari users, user_profile, consumption_record, categories, dan articles. Desain database aplikasi pemantauan konsumsi gula harian dapat dilihat pada Gambar 4.17.



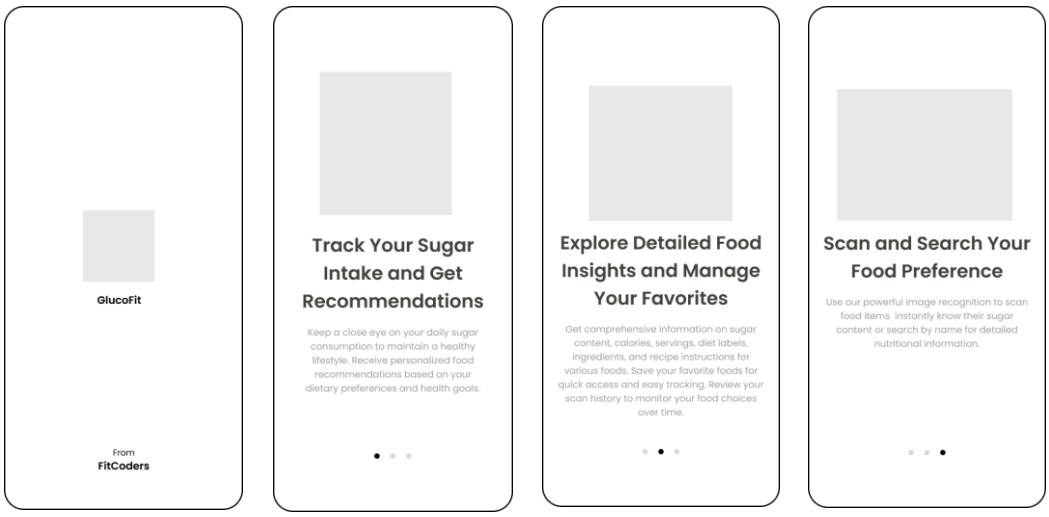

Gambar 4. 17 Rancangan Database

### 4.4.4 *Wireframe*

Berikut rancangan antarmuka yang telah dibuat untuk aplikasi pemantauan konsumsi gula harian.

a. *Splash Screen* dan Halaman Onboarding

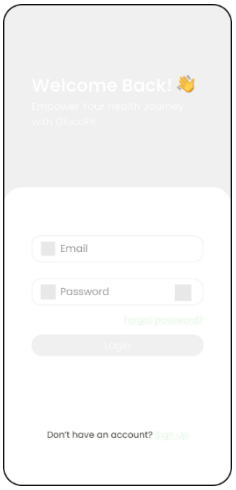
*Splash Screen* berisikan logo aplikasi yang muncul ketika pengguna membuka aplikasi. Setelah beberapa detik aplikasi akan menampilkan halaman onboarding. Halaman onboarding berisikan gambar dan teks yang berfungsi untuk mengenalkan aplikasi pada pengguna.



Gambar 4. 18 *Wireframe Splash Screen* dan Onbording

b. Halaman Login

Pada halaman ini terdapat kalimat selamat datang yang akan diletakan dibagian tengah sisi atas . Selain itu, pada sisi bawah terdapat input field untuk username dan password serta button login dibagian bawahnya.




Gambar 4. 19 *Wireframe* Login

c. Halaman Registrasi

Pada halaman ini terdapat kalimat selamat hallo yang akan diletakan dibagian tengah sisi atas . Selain itu, pada sisi bawah terdapat input field untuk username,email dan password serta button register dibagian bawahnya.

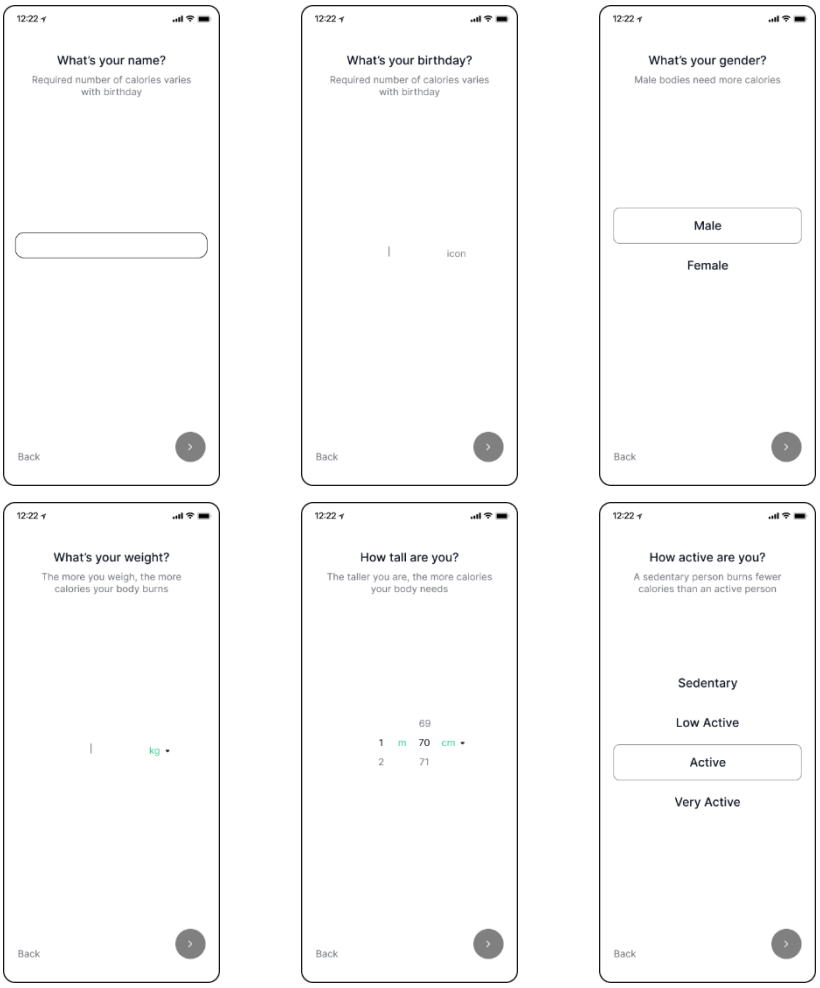



Gambar 4. 20 *Wireframe* Register

d. Halaman Informasi Penguna

Pada halaman ini berisi beberapa kolom untuk mengisi data dasar pengguna, seperti nama, tanggal lahir, jenis kelamin, berat badan, tinggi badan, dan tingkat aktivitas. Setiap kolom

memiliki input field yang jelas, dan pengguna dapat melanjutkan dengan menekan tombol "Next" setelah mengisi setiap bagian.




Gambar 4. 21 *Wireframe* Informasi Penguna

e. Halaman Home

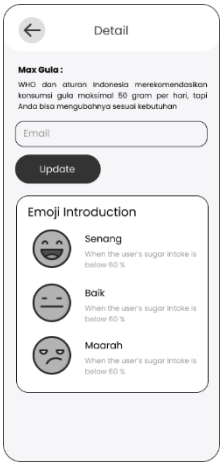
Halaman utama aplikasi ini menampilkan level gula makanan dan status pengguna. Pengguna dapat memilih untuk menghitung gula makanan secara otomatis atau mengatur jumlah gula secara manual melalui foto. Terdapat juga informasi tentang pola makan sehat dan edukasi gula.




Gambar 4. 22 *Wireframe* Home

f. Halaman Detail Informasi

Halaman detail informasi ini menampilkan batas konsumsi gula harian sesuai rekomendasi WHO dan regulasi di Indonesia, yaitu maksimal 50 gram per hari. Namun, pengguna diberikan kebebasan untuk menyesuaikan batas tersebut sesuai dengan kebutuhan pribadi melalui kolom input yang tersedia. Selain itu, halaman ini juga memperkenalkan sistem emoji yang merepresentasikan kondisi konsumsi gula pengguna, seperti emoji ‘Senang’ untuk konsumsi di bawah 60%, ‘Baik’ untuk tingkat sedang untuk 60% - 90% , dan ‘Marah’ jika konsumsi melebihi ambang batas atau lebih dari 100% . Fitur ini dirancang untuk memberikan pengalaman visual yang mudah dipahami terkait status konsumsi gula harian pengguna.




Gambar 4. 23 *Wireframe* Detail Informasi

g. Halaman Detail Artikel

Pada halaman ini terdapat gambar, judul dan deskripsi. Pengguna dapat membaca informasi mengenai artikel dan tips yang mereka pilih.




Gambar 4. 24 *Wireframe* Detail Artikel

h. Halaman Pencatatan Manual

Halaman ini memungkinkan pengguna untuk menginput data secara manual dengan mengunggah gambar makanan. Pengguna dapat menambahkan nama makanan, jumlah gula, dan jumlah item yang dikonsumsi. Terdapat juga opsi untuk menambah foto langsung dari kamera.




Gambar 4. 25 *Wireframe* Pencatatan Manual

i. Halaman Pencatatan Otomatis

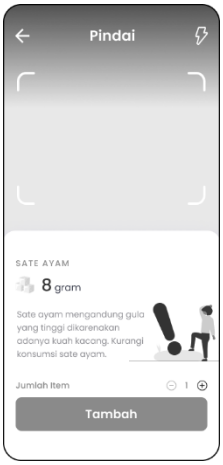
Halaman ini memungkinkan pengguna untuk menghitung kandungan gula makanan secara otomatis dengan memindai gambar makanan. Pengguna dapat mengambil foto dengan kamera atau memilih gambar dari galeri. Terdapat petunjuk untuk memotret makanan.




Gambar 4. 26 *Wireframe* Pencataan Otomatis

j. Halaman Hasil Pencatatan Otomatis

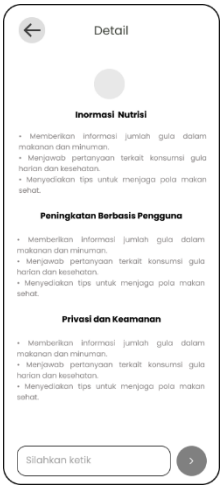
Halaman ini menampilkan hasil pemindaian makanan yang mengidentifikasi jumlah gula dalam makanan. Pengguna dapat melihat informasi tentang makanan dan jumlah gula yang dipindai, serta penjelasan dari hasil pindaian tersebut. Terdapat opsi untuk menambah jumlah item yang dikonsumsi.




Gambar 4. 27 *Wireframe* Hasil Pencatatan Otomatis

k. Halaman Chat Bot

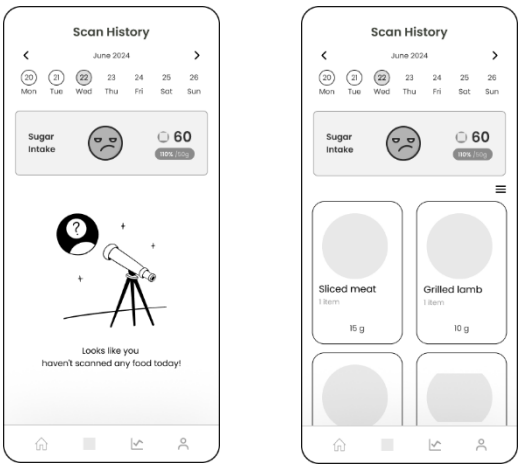
Pada halaman ini, pengguna dapat mengetikkan pesan pada kolom yang tersedia. Setelah selesai mengetik pesan, pengguna dapat menekan tombol "Kirim" untuk mengirim pesan tersebut. Sistem kemudian akan memproses dan merespons pesan yang dikirim oleh pengguna.




Gambar 4. 28 *Wireframe* Chat Bot

l. Halaman Laporan

Pada halaman ini, pengguna dapat melihat riwayat pemindaian makanan berdasarkan tanggal yang dipilih, lengkap dengan informasi level gula makan. Jika tidak ada data pada tanggal tersebut, akan muncul pesan yang menyatakan "Looks like you haven’t scanned any food today!" di bagian tengah layar. Namun, jika ada makanan yang dipindai, hasilnya akan ditampilkan dengan nama makanan, jumlah item, kandungan gula dalam gram, serta informasi level gula makan yang tercatat pada hari tersebut.

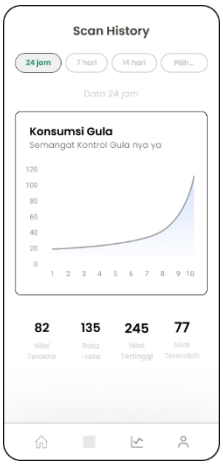



Gambar 4. 29 *Wireframe* Laporan

m. Halaman Statistik

Pada halaman ini, pengguna dapat melihat grafik konsumsi gula darah berdasarkan pilihan periode waktu, yaitu 24 jam, 7 hari, 30 hari, atau memilih tanggal tertentu. Terdapat informasi mengenai nilai terakhir, rata-rata, nilai tertinggi, dan nilai terendah yang tercatat.

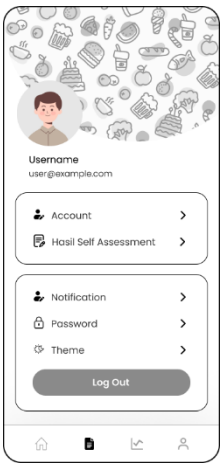
Grafik ini memberikan gambaran tentang perubahan kadar gula makanan, disertai dengan pesan motivasi "Semangat Kontrol Gula nya ya" untuk mendorong pengguna menjaga kadar gula darah tetap stabil.




Gambar 4. 30 *Wireframe* Statistik

n. Halaman Profile

Pada halaman ini, pengguna dapat mengakses informasi akun mereka, hasil informasi pengguna, dan terdapat opsi untuk melakukan perubahan akun dan password. Pengguna juga dapat keluar dari akun dengan menekan tombol "Log Out" di bagian bawah.

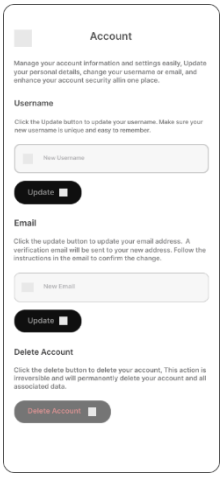



Gambar 4. 31 *Wireframe* Profile

o. Halmanan Account

Halaman ini berisi beberapa kolom untuk memperbarui informasi akun pengguna, seperti nama pengguna dan alamat email. Setiap kolom dilengkapi dengan input field yang jelas, dan pengguna dapat melanjutkan dengan menekan tombol "*Update*" setelah mengisi setiap

bagian. Terdapat juga tombol "Delete Account" di bagian bawah untuk menghapus akun secara permanen bersama dengan semua data terkait.




Gambar 4. 32 *Wireframe* Account

p. Halaman Reset Password

Pada halaman ini, pengguna dapat memperbarui password mereka. Pengguna diminta untuk memasukkan password lama pada kolom pertama, kemudian memasukkan password baru pada kolom kedua, dan mengkonfirmasi password baru tersebut pada kolom ketiga. Setelah memasukkan password dengan benar, pengguna dapat menekan tombol *Update* untuk menyimpan perubahan password mereka.




Gambar 4. 33 *Wireframe* Password

q. Halaman Hasil Informasi Pengguna

Halaman ini menampilkan hasil penilaian diri pengguna, yang mencakup informasi tentang konsumsi gula harian, konsumsi kalori harian, dan tanggal assessment. Hasil tersebut berdasarkan jawaban yang diberikan dalam penilaian mandiri sebelumnya. Selain

itu, halaman ini juga mencantumkan data pribadi pengguna, termasuk nama, tanggal lahir, jenis kelamin, berat badan, tinggi badan, dan tingkat aktivitas, dengan kolom input yang dapat diisi sesuai kebutuhan.




Gambar 4. 34 *Wireframe* hasil Informasi Pengguna

## 4.5 Pegembangan

Sistem ini dikembakan berdasarkan rancangan sistem yang telah dijabarkan pada tahap sebelumnya. Pegembangan ini dilakukan menggunakan bahasa pemrograman Kotlin dengan editor android studio untuk pembuatan frontend dan menggunakan express.js untuk pembuatan backend.

### 4.5.1 Configurasi api

Pada Gambar 4.35, dapat dilihat bahwa pada bagian ini digunakan untuk menangani permintaan API *https://express-api-632437931482.asia-southeast2.run.app/api/* menggunakan Retrofit dan OkHttpClient. Fungsi getApiService() memastikan hanya ada satu instance OkHttpClient dengan memeriksa apakah klien sudah dibuat; jika belum, klien dibuat melalui *createClient().* Retrofit dikonfigurasi dengan base URL dan dikembalikan sebagai instance *ApiService* untuk melakukan permintaan API. Fungsi *createClient()* mengatur OkHttpClient dengan *HttpLoggingInterceptor* untuk mencatat request dan response, AuthInterceptor untuk autentikasi berbasis *UserPreference*, serta timeout koneksi, pembacaan, dan penulisan yang masing-masing ditetapkan selama 60 detik. Dengan konfigurasi ini, aplikasi dapat mengakses API dengan logging dan autentikasi yang aman.

```
companion object {
    private var client: OkHttpClient? = null

    fun getApiService(context: Context, preferences: UserPreference):
ApiService {
        if (client == null) {
            client = createClient(context.applicationContext, preferences)
        }
        val retrofit = Retrofit.Builder()
            .baseUrl("https://express-api-632437931482.asia-
southeast2.run.app/api/")
            .addConverterFactory(GsonConverterFactory.create())
            .client(client!!)
            .build()
        return retrofit.create(ApiService::class.java)
    }

    private fun createClient(context: Context, preferences:
UserPreference): OkHttpClient {
        val loggingInterceptor =
HttpLoggingInterceptor().setLevel(HttpLoggingInterceptor.Level.BODY)

        return OkHttpClient.Builder()
            .addInterceptor(loggingInterceptor)
            .addInterceptor(AuthInterceptor(context.applicationContext,
preferences))
            .connectTimeout(60, TimeUnit.SECONDS)
            .readTimeout(60, TimeUnit.SECONDS)
            .writeTimeout(60, TimeUnit.SECONDS)
            .build()
    }
}
```

Gambar 4. 35 Configurasi Api

### 4.5.2 *Splash Screen*

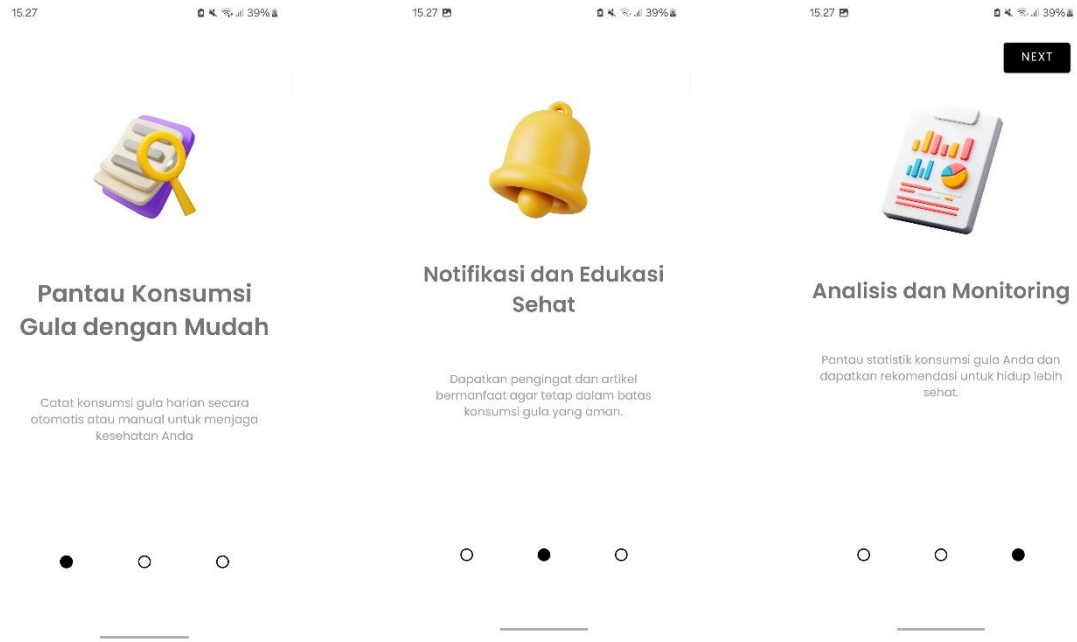
Halaman merupakan tampilan dari *Splash Screen* pada aplikasi yang muncul sesaat ketika aplikasi pertama kali dibuka. Halaman ini menampilkan logo dari aplikasi selama beberapa detik. Implementasi *Splash Screen* dapat dilihat pada Gambar 4.36.




Gambar 4. 36 *Splash Screen*

### 4.5.3 Halaman Onboarding

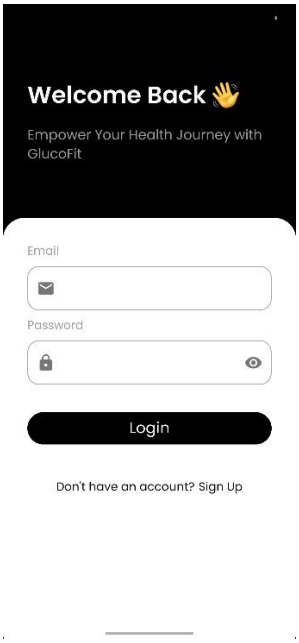
merupakan tampilan dari halaman onboarding pada aplikasi yang muncul setelah *Splash Screen* dan pengguna belum masuk ke dalam akun yang dimiliki. Halaman ini berfungsi untuk mengenalkan fitur aplikasi kepada pengguna. Implementasi halaman Onbording dapat dilihat pada Gambar 4.37.




Gambar 4. 37 Onbording

### 4.5.4 Halaman Login

Halaman login ini merupakan fitur utama pada aplikasi GlucoFit. Pada halaman ini, pengguna dapat memasukkan alamat email dan kata sandi untuk mengakses aplikasi. Terdapat tombol "Login" yang memungkinkan pengguna melanjutkan, serta opsi "Sign Up" bagi pengguna baru. Tampilan sederhana dengan kolom input yang jelas memudahkan pengguna untuk melakukan login. Implementasi halaman login dapat dilihat pada Gambar 4.38.




Gambar 4. 38 Login

Pada Gambar 4.39, dapat dilihat bahwa pada bagian kode ini terdapat pengaturan untuk tombol login dan tombol register. fungsi *setupLoginButton()* mengatur tombol login. Ketika tombol login ditekan, aplikasi akan mengambil nilai dari input email dan password. Jika salah satu kosong, maka akan muncul pesan error menggunakan *showCustomErrorDialog().* Jika keduanya terisi, indikator loading ditampilkan dengan *showLoading(true),* dan fungsi *loginViewModel.postLogin(email, password)* akan dijalankan untuk mengirimkan data login.

Untuk tombol register, ketika ditekan, aplikasi akan membuka layar registrasi (RegisterActivity) menggunakan *startActivity()*, kemudian menutup aktivitas login saat ini dengan *finish().*

```
    private fun setupLoginButton() {
        binding.loginButton.setOnClickListener {
            val email = binding.emailEditText.text.toString()
            val password = binding.passwordEditText.text.toString()

            if (email.isEmpty() || password.isEmpty()) {
                showCustomErrorDialog("Email atau Password tidak boleh
kosong.")
            } else {
```

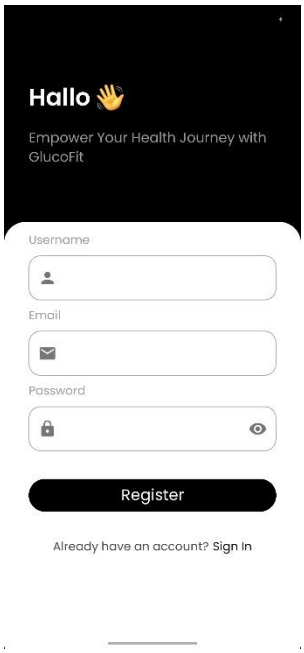
```
                showLoading(true)
                loginViewModel.postLogin(email, password)
            }
        }

        binding.registerButton.setOnClickListener {
            startActivity(Intent(this, RegisterActivity::class.java))
            finish()
        }
    }
```

Gambar 4. 39 *Code* Halaman Login

### 4.5.5 Halaman Register

Halaman registrasi ini merupakan bagian dari aplikasi GlucoFit, yang memungkinkan pengguna untuk membuat akun baru. Pada halaman ini, pengguna diminta untuk mengisi kolom username, email, dan kata sandi. Terdapat tombol "Register" yang akan mengkonfirmasi pendaftaran setelah data diisi. Jika pengguna sudah memiliki akun, mereka dapat memilih opsi "Sign In" untuk masuk ke aplikasi. Implementasi halaman pendaftaran dapat dilihat pada Gambar 4.40.




Gambar 4. 40 Register

Pada Gambar 4.41, dapat dilihat bahwa pada bagian kode ini , terdapat dua fungsi utama. Fungsi pertama, *setOnClickListener* pada signinButton, digunakan untuk menangani klik tombol "Sign In". Ketika tombol diklik, aplikasi akan membuka LoginActivity menggunakan Intent.

Fungsi kedua, *postText()*, digunakan untuk menangani proses pendaftaran pengguna. Saat tombol "Login" ditekan, sistem mengambil input dari pengguna, yaitu nama, email, dan

password. Jika semua input terisi, maka data dikirim ke ViewModel melalui metode *postRegister()*. Jika ada input yang kosong, aplikasi akan menampilkan pesan kesalahan untuk mengingatkan pengguna agar melengkapi semua kolom yang diperlukan.

```
    // OnClickListener untuk button signin
    binding.signinButton.setOnClickListener {
        startActivity(Intent(this, LoginActivity::class.java))
    }

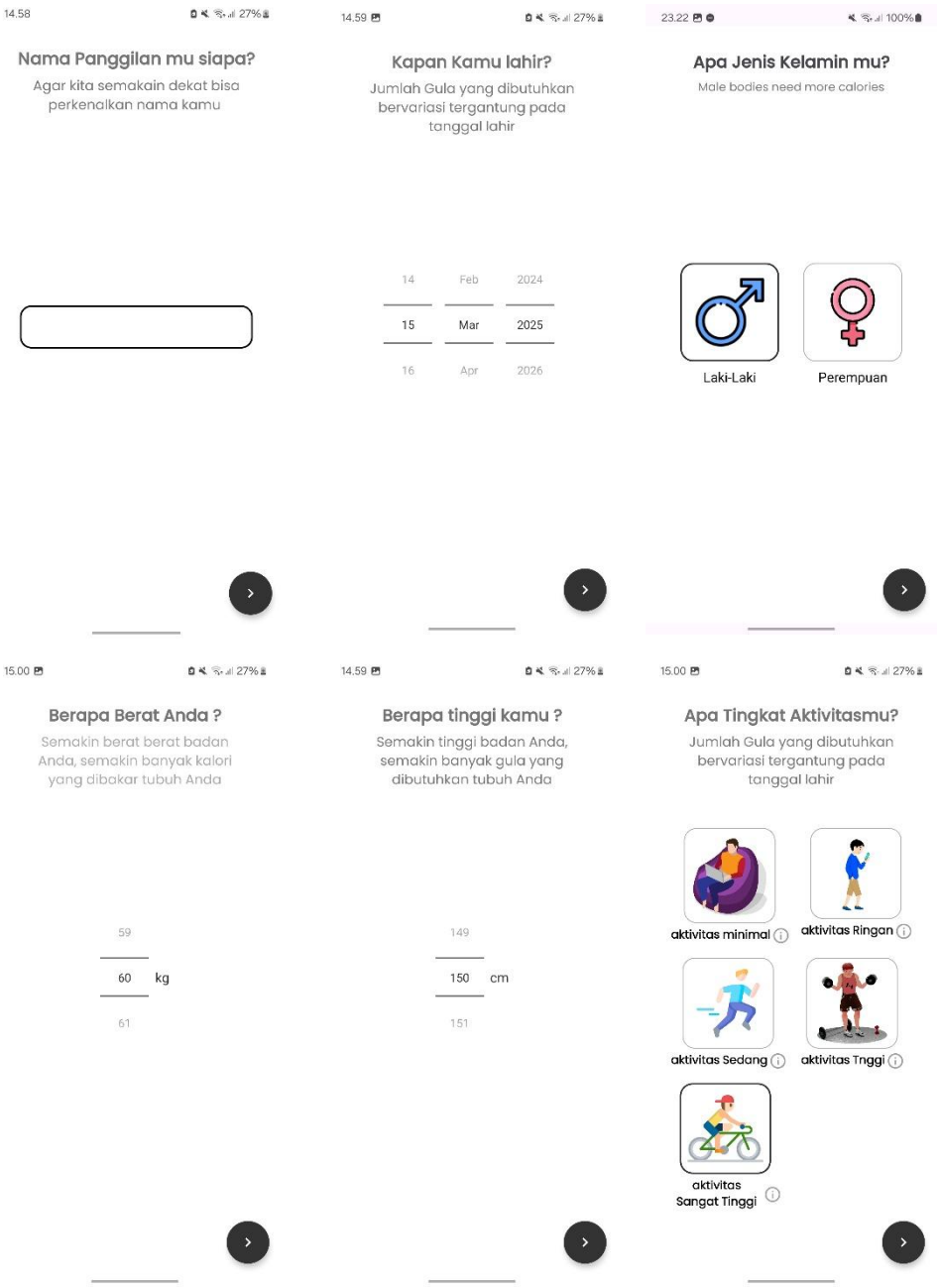
    // Fungsi untuk meng-handle login dan registrasi
    private fun postText() {
        binding.loginButton.setOnClickListener {
            val userName = binding.nameEditText.text.toString()
            val email = binding.emailEditText.text.toString()
            val password = binding.passwordEditText.text.toString()

            if    (userName.isNotEmpty()    &&    email.isNotEmpty()    &&
password.isNotEmpty()) {
                // Memanggil method postRegister pada ViewModel
                registerViewModel.postRegister(userName, email, password)
            } else {
                showErrorMessage("Please fill in all fields.")
            }
        }
    }
```

Gambar 4. 41 *Code* Halaman Register

### 4.5.6 Halaman Infomasi Penguna

Halaman informasi pengguna dalam aplikasi GlucoFit dirancang untuk mengumpulkan data pengguna yang relevan guna menghitung kebutuhan konsumsi gula harian secara lebih akurat. Pada halaman ini, pengguna diminta untuk mengisi beberapa informasi, termasuk nama panggilan, tanggal lahir, jenis kelamin, berat badan, tinggi badan, dan tingkat aktivitas fisik. Data ini berperan penting dalam menentukan batas konsumsi gula harian yang disesuaikan dengan karakteristik individu pengguna. Setiap informasi dimasukkan melalui antarmuka yang interaktif, dengan opsi navigasi untuk berpindah ke langkah berikutnya. Implementasi dari halaman informasi pengguna ini dapat dilihat pada Gambar 4.42.




Gambar 4. 42 Informasi Penguna

Pada Gambar 4.43, dapat dilihat bahwa pada bagian kode ini berfungsi untuk menangani navigasi dan penyimpanan data dalam proses asesmen pengguna. Setiap halaman memiliki tombol "Next" yang akan menyimpan input pengguna ke dalam assesmentViewModel sebelum berpindah ke halaman berikutnya. Pada halaman nama, sistem mengambil input nama pengguna dan menyimpannya ke *assesmentViewModel.name*, lalu berpindah ke halaman usia. Selanjutnya, pada halaman usia, tanggal lahir dipilih melalui DatePicker, kemudian disimpan dalam *assesmentViewModel.birthday* sebelum berpindah ke halaman tinggi badan. Pada tahap berikutnya, tinggi badan diambil dari centimeterPicker dan disimpan dalam

*assesmentViewModel.height*, lalu pengguna diarahkan ke halaman berat badan. Begitu juga dengan berat badan, nilainya diambil dari meterPicker dan disimpan di *assesmentViewModel.weight*, lalu navigasi dilanjutkan ke halaman gender. Jika pengguna telah memilih gender, nilai tersebut disimpan dalam *assesmentViewModel.gender*, dan sistem akan berpindah ke halaman aktivitas. Jika gender belum dipilih, akan muncul pesan peringatan agar pengguna memilihnya terlebih dahulu. Setelah semua data terkumpul, fungsi *postdata()* digunakan untuk menggabungkan nama, gender, tanggal lahir, berat badan, tinggi badan, dan tingkat aktivitas pengguna, kemudian mengirimkan data tersebut ke *assesmentViewModel.postUserData()* untuk diproses lebih lanjut.

```
    // Tombol Next halaman usia
    val nextButton: ImageButton = view.findViewById(R.id.nextButton)
    nextButton.setOnClickListener {
        val name = usernameEditText.text.toString().trim()
        assesmentViewModel.name.value = name

findNavController().navigate(R.id.action_nameFragment_to_AgeFragmen
t)
    }

    // Tombol Next halaman tinggi badan
    nextButton.setOnClickListener {
        val day = datePicker.dayOfMonth
        val month = datePicker.month + 1 // Bulan dimulai dari 0, jadi
tambahkan 1
        val year = datePicker.year
        val selectedDate = "$year-$month-$day"
        assesmentViewModel.birthday.value = selectedDate

findNavController().navigate(R.id.action_ageFragment_to_heightFragm
ent)
    }

    // Tombol Next halaman berat badan
    nextButton.setOnClickListener {
        val selectedHeight = centimeterPicker.value.toString()
        assesmentViewModel.height.value = selectedHeight

findNavController().navigate(R.id.action_heightFragment_to_weightFr
agment)
    }

    // Tombol Next ke halaman gender
    nextButton.setOnClickListener {
        val selectedHeight = meterPicker.value.toString()
        assesmentViewModel.weight.value = selectedHeight

findNavController().navigate(R.id.action_weightFragment_to_genderFr
agment)
    }

    // Tombol Next aktivitas
    nextButton.setOnClickListener {
```

```
        if (selectedGender != null) {
            assesmentViewModel.gender.value = selectedGender

findNavController().navigate(R.id.action_genderFragment_to_activity
Fragment)
        } else {
            // Tampilkan pesan jika gender belum dipilih
            Toast.makeText(requireContext(),  "Please  select  your
gender", Toast.LENGTH_SHORT).show()
        }
    }

    fun postData() {
        val nama = assesmentViewModel.name.value ?: ""
        val jenisKelamin = assesmentViewModel.gender.value ?: ""
        val tanggalLahir = assesmentViewModel.birthday.value ?: ""
        val beratBadan = assesmentViewModel.weight.value ?: ""
        val tinggiBadan = assesmentViewModel.height.value ?: ""
        val aktivitas = assesmentViewModel.activityLevel.value ?: ""

        assesmentViewModel.postUserData(nama,          jenisKelamin,
tanggalLahir, beratBadan, tinggiBadan, aktivitas)
    }
```

Gambar 4. 43 *Code* Halaman Informasi Penguna

### 4.5.7 Halaman Home

Halaman utama aplikasi GlucoFit menyambut pengguna dengan informasi tentang tingkat gula darah. Indikator "Sugar level" menunjukkan tiga tingkatan warna: hijau (Senang) untuk konsumsi gula yang sehat, kuning (Baik) untuk konsumsi gula yang perlu perhatian, dan merah (Sedih) ketika konsumsi gula melebihi batas yang disarankan. Emotikon juga menampilkan status pengguna berdasarkan konsumsi gula: "Senang" jika mencapai 60%, "Baik" jika antara 60%-90%, dan "Sedih" jika lebih dari 100%. Pengguna dapat memindai makanan dengan dua opsi: "Hitung otomatis" untuk foto makanan atau "Atur Jumlahnya sendiri" untuk memasukkan nama makanan dan kandungan gula. Selain itu, terdapat juga "Scan Bar*Code*" yang penguna bisa nemapilkan nutrisi dari produk makanan atau minuman, tersedia artikel edukasi mengenai pengelolaan konsumsi gula. Implementasi halaman utama dapat dilihat pada Gambar 4.44.




Gambar 4. 44 Home

Pada Gambar 4.45, dapat dilihat bahwa pada bagian kode ini memiliki fungsi *setupClickListeners()* bertanggung jawab untuk menangani aksi ketika pengguna menekan berbagai tombol pada aplikasi. Setiap tombol akan membuka aktivitas yang sesuai dengan fungsinya. Jika pengguna menekan tombol *countAutomation*, maka aplikasi akan membuka *ScannerActivity*, yang kemungkinan digunakan untuk pemindaian otomatis. Tombol *countManual* akan membuka *ManualActivity*, yang mungkin berfungsi untuk pencatatan konsumsi gula secara manual. Tombol *dailySugarIntakeCard.icInfo* akan membuka *DetailActivity*, yang mungkin menampilkan rincian konsumsi gula. Sedangkan tombol *chatbot* akan membuka *ChatActivity*, yang bisa digunakan untuk berinteraksi dengan chatbot, dan tombol *countLabel* akan membuka *LabelActivity*, yang kemungkinan digunakan untuk mengenali label makanan.

Fungsi *updateUI()* bertugas memperbarui tampilan berdasarkan jumlah konsumsi gula harian pengguna. Jika konsumsi gula melebihi 90% dari batas harian, tampilan berubah menjadi merah dengan ikon dan latar yang menandakan peringatan tinggi, serta emoji yang menunjukkan ekspresi marah. Jika konsumsi berada di antara 60% dan 90%, tampilan berubah menjadi kuning dengan indikator yang lebih netral dan ekspresi wajah "biasa". Jika konsumsi masih dalam batas aman di bawah 60%, tampilan tetap hijau dengan ikon yang menunjukkan kondisi sehat dan ekspresi senang. Selain memperbarui tampilan, fungsi ini juga menampilkan jumlah gula yang dikonsumsi dalam gram, persentase konsumsi terhadap batas harian, serta batas konsumsi maksimum dalam satuan gram. Setelah memperbarui tampilan, fungsi ini

memeriksa apakah perlu mengirim notifikasi kepada pengguna terkait konsumsi gula dengan memanggil *checkAndSendNotifications()*, serta mereset status notifikasi jika dibutuhkan melalui *resetNotificationStatusIfNeeded().*

```
    // Menyiapkan OnClickListener untuk setiap tombol di UI
    private fun setupClickListeners() {
        // Tombol untuk membuka ScannerActivity
        binding.countAutomation.setOnClickListener {
            startActivity(Intent(activity,
ScannerActivity::class.java))
        }

        // Tombol untuk membuka ManualActivity
        binding.countManual.setOnClickListener {
            startActivity(Intent(activity,
ManualActivity::class.java))
        }

        // Tombol untuk membuka DetailActivity
        binding.dailySugarIntakeCard.icInfo.setOnClickListener {
            startActivity(Intent(activity,
DetailActivity::class.java))
        }

        // Tombol untuk membuka ChatActivity
        binding.chatbot.setOnClickListener {
            startActivity(Intent(activity,
ChatActivity::class.java))
        }

        // Tombol untuk membuka LabelActivity
        binding.countLabel.setOnClickListener {
            startActivity(Intent(activity,
LabelActivity::class.java))
        }
    }

    // Mengupdate UI berdasarkan konsumsi gula dan persentase konsumsi
    private fun updateUI(sugarConsumed: Int, dailyLimit: Int,
percentageConsumed: Int) {
        when {
            // Jika persentase konsumsi gula lebih dari 90%
            percentageConsumed > 90 -> {

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.
rectangle_filled_red)

binding.dailySugarIntakeCard.line.setBackgroundResource(R.color.lin
e_yellow)

binding.dailySugarIntakeCard.icon.setImageResource(R.drawable.ic_ho
me_icon_red)

binding.dailySugarIntakeCard.iconContaint.setBackgroundResource(R.d
rawable.ic_background_red)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawab
le.shape_red)
```

```
binding.dailySugarIntakeCard.imageEmosi.setImageResource(R.drawable
.angry)
                binding.dailySugarIntakeCard.emosi.text = "Merah" //
Menampilkan status emosi "Merah"
            }
            // Jika persentase konsumsi gula lebih dari 60%
            percentageConsumed > 60 -> {

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.
rectangle_filled_yellow)

binding.dailySugarIntakeCard.line.setBackgroundResource(R.color.lin
e_red)

binding.dailySugarIntakeCard.icon.setImageResource(R.drawable.ic_ho
me_icon_yellow)

binding.dailySugarIntakeCard.iconContaint.setBackgroundResource(R.d
rawable.ic_background_yellow)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawab
le.shape_yellow)

binding.dailySugarIntakeCard.imageEmosi.setImageResource(R.drawable
.good)

binding.dailySugarIntakeCard.tvSugarIntakePercentage.setTextColor(C
ontextCompat.getColor(requireContext(), R.color.black))

binding.dailySugarIntakeCard.tvSugarIntakeMax.setTextColor(ContextC
ompat.getColor(requireContext(), R.color.black))
                binding.dailySugarIntakeCard.emosi.text = "Biasa" //
Menampilkan status emosi "Biasa"
            }
            // Jika persentase konsumsi gula di bawah 60%
            else -> {

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.
rectangle_filled)

binding.dailySugarIntakeCard.iconContaint.setBackgroundResource(R.d
rawable.ic_background_green)

binding.dailySugarIntakeCard.icon.setImageResource(R.drawable.ic_ho
me_icon)

binding.dailySugarIntakeCard.line.setBackgroundResource(R.color.lin
e_green)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawab
le.shape_green)

binding.dailySugarIntakeCard.imageEmosi.setImageResource(R.drawable
.happy)
                binding.dailySugarIntakeCard.emosi.text  =  "Senang"
// Menampilkan status emosi "Senang"
            }
        }
```

```
        // Menampilkan data konsumsi gula, persentase, dan batas
konsumsi gula
        binding.dailySugarIntakeCard.sugarConsumed.text            =
sugarConsumed.toString()
        binding.dailySugarIntakeCard.tvSugarIntakePercentage.text  =
"$percentageConsumed%"
        binding.dailySugarIntakeCard.tvSugarIntakeMax.text         =
"/${dailyLimit}g"

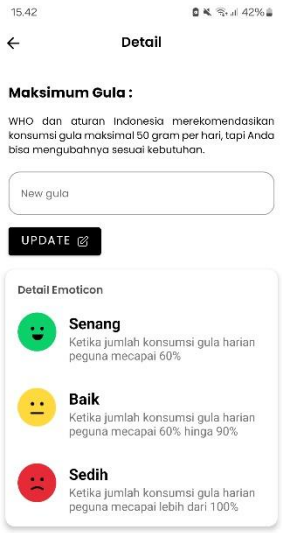
        // Memeriksa dan mengirim notifikasi jika diperlukan
        checkAndSendNotifications(sugarConsumed,         dailyLimit,
percentageConsumed)

        // Mereset status notifikasi jika sudah dipenuhi
        resetNotificationStatusIfNeeded(sugarConsumed, dailyLimit)
    }
```

Gambar 4. 45 *Code* Halaman Home

### 4.5.8 Halaman Detail Informasi

Halaman detail ini merupakan fitur yang memungkinkan pengguna untuk memantau dan mengubah batas konsumsi gula harian mereka. Di bagian atas, terdapat informasi tentang rekomendasi WHO dan aturan Indonesia yang menyarankan batas konsumsi gula maksimal 50 gram per hari, namun pengguna dapat menyesuaikan jumlah tersebut sesuai kebutuhan. Pengguna dapat memasukkan jumlah gula yang baru pada kolom "New gula" dan mengklik tombol "*Update*" untuk memperbarui pengaturan. Di bawahnya, terdapat tiga emotikon dengan deskripsi yang menggambarkan status konsumsi gula, yaitu: "Senang" (untuk konsumsi gula hingga 60%), "Baik" (untuk konsumsi gula 60%-90%), dan "Sedih" (untuk konsumsi gula yang melebihi 100%). Implementasi halaman detail ini dapat dilihat pada Gambar 4.46.




Gambar 4. 46 Detail Informasi

### 4.5.9 Halaman Edukasi

Halaman edukasi ini merupakan bagian dari halaman utama aplikasi Pencatan konsumsi gula harian yang menyediakan artikel dalam empat kategori: Pola Makan Sehat, Fakta dan Edukasi tentang Gula, Motivasi dan Kebiasaan Sehat, dan Informasi Ilmiah dan Kesehatan. Artikel-artikel ini memberikan informasi penting mengenai pengelolaan gula dan gaya hidup sehat. Implementasi halaman edukasi dapat dilihat pada Gambar 4.47.




Gambar 4. 47 Edukasi

### 4.5.10 Halaman Detail Edukasi

Halaman detail edukasi dalam aplikasi GlucoFit dirancang untuk menampilkan informasi secara lebih mendalam dalam format yang mudah dibaca oleh pengguna. Halaman ini menampilkan teks dengan tata letak yang terstruktur, disertai dengan gambar ilustratif yang mendukung isi informasi. Pengguna dapat mengakses konten secara interaktif dan menutup halaman jika telah selesai membaca. Implementasi dari halaman detail edukasi ini dapat dilihat pada Gambar 4.48.



Gambar 4. 48 Detail Edukasi

### 4.5.11 Halaman Pencatatan Manual

Halaman pencatatan manual memungkinkan pengguna untuk menambahkan makanan secara manual dengan mengunggah gambar, serta memasukkan nama makanan dan jumlah gula berdasarkan pengetahuan pengguna. Fitur ini tidak dapat mendeteksi komposisi makanan melalui pemindaian label atau analisis otomatis, sehingga pengguna harus mengisi informasi secara manual. Implementasi halaman input manual dapat dilihat pada Gambar 4.49.




Gambar 4. 49 Pencatatan Manual

Pada Gambar 4.50, dapat dilihat bahwa pada bagian kode ini bertanggung jawab atas interaksi pengguna dalam aplikasi, khususnya dalam pemrosesan gambar dan input data makanan. Fungsi *setupListeners()* mengatur berbagai tombol untuk menangani tindakan pengguna. Tombol *galleryButton* digunakan untuk membuka galeri guna memilih gambar, sementara *cameraButton* meluncurkan kamera setelah memeriksa izin yang diperlukan. Tombol *btnIncrement* dan *btnDecrement* memungkinkan pengguna menyesuaikan jumlah porsi makanan. Saat tombol *analyzeButton* ditekan, aplikasi mengambil nama makanan, jumlah gula per porsi, dan jumlah porsi yang dipilih, lalu menghitung total konsumsi gula. Jika ada gambar yang dipilih atau diambil, gambar tersebut diproses dan bersama dengan data makanan dikirim ke server, sebelum akhirnya pengguna diarahkan ke MainActivity.

Selain itu, fungsi *launchCamera()* menangani pengambilan gambar dengan menyimpan hasilnya ke dalam file dan mengatur izin akses. Dua objek *launcherIntentGallery* dan *launcherIntentCamera* digunakan untuk menangani hasil pemilihan gambar dari galeri atau

kamera, menampilkan gambar yang dipilih, serta memprosesnya menggunakan *uCropUtils()* untuk pemotongan gambar. Jika terjadi kesalahan dalam pengambilan atau pemrosesan gambar, aplikasi akan menampilkan pesan error kepada pengguna.

```
    private fun setupListeners() {
        // Membuka galeri untuk memilih gambar
        binding.galleryButton.setOnClickListener {
            val intent = Intent(Intent.ACTION_GET_CONTENT)
            intent.type = "image/*"
            val chooser = Intent.createChooser(intent, null)
            launcherIntentGallery.launch(chooser)
        }

        // Menangani izin dan meluncurkan kamera
        binding.cameraButton.setOnClickListener {
            if (checkCameraPermission()) {
                launchCamera()
            } else {
                requestCameraPermission()
            }
        }

        // Menambah nilai porsi
        binding.btnIncrement.setOnClickListener {
            currentValue++
            binding.tvValue.text = currentValue.toString()
        }

        // Mengurangi nilai porsi
        binding.btnDecrement.setOnClickListener {
            if (currentValue > 1) {
                currentValue--
            }
            binding.tvValue.text = currentValue.toString()
        }

        // Mengirimkan data untuk analisis
        binding.analyzeButton.setOnClickListener {
            val foodName = binding.usernameEditText.text.toString()
            val                    sugarAmount                    =
binding.FoodEditText.text.toString().toDoubleOrNull() ?: 0.0
            val                      portion                      =
binding.tvValue.text.toString().toDoubleOrNull() ?: 0.0
            val totalSugar = sugarAmount * portion

            // Mengunggah gambar dan data analisis
            croppedImageUri?.let { uri ->
                val imageFile = File(uri.path!!)
                val imageBody = MultipartBody.Part.createFormData(
                    "gambar",                        imageFile.name,
imageFile.asRequestBody("image/jpeg".toMediaTypeOrNull())
                )
                val                foodNameRequest                =
foodName.toRequestBody("text/plain".toMediaTypeOrNull())
                val              sugarAmountRequest               =
totalSugar.toString().toRequestBody("text/plain".toMediaTypeOrNull(
))
```

```
                val                    servingSize                    =
portion.toString().toRequestBody("text/plain".toMediaTypeOrNull())
                val                    inputMethod                    =
"manual".toRequestBody("text/plain".toMediaTypeOrNull())

                scanViewModel.uploadScanImage(imageBody,
foodNameRequest, sugarAmountRequest, servingSize, inputMethod)
                startActivity(Intent(this,
MainActivity::class.java))
                Toast.makeText(this,     "Berhasil     Menambahkan",
Toast.LENGTH_SHORT).show()
            }
        }
    }

    // Menangani hasil dari galeri
    private val launcherIntentGallery = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
    ) { result ->
        if (result.resultCode == RESULT_OK) {
            val selectedImg = result.data?.data
            selectedImg?.let { uri ->
                initImageUri = uri
                showImage()
                uCropUtils(uri)
            } ?: showToast(getString(R.string.image_get_uri_error))
        }
    }

    // Menangani hasil dari kamera
    private val launcherIntentCamera = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
    ) { result ->
        if (result.resultCode == RESULT_OK) {
            initImageUri = photoUri
            showImage()
            uCropUtils(photoUri)
        } else {
            showToast(getString(R.string.image_capture_error))
        }
    }
```

Gambar 4. 50 *Code* Halaman Pencatatan Manual

### 4.5.12 Halaman Pencatatan Otomatis

Pada halaman pencatatan otomatis memungkinkan pengguna untuk memotret makanan yang akan dianalisis menggunakan model machine learning. Model ini, yang dikembangkan oleh rekan peneliti, memberikan output berupa nama makanan dan jumlah gula yang terkandung. Namun, model ini hanya dapat menghitung makanan dalam porsi utuh. Jika makanan yang dipotret hanya berupa sebagian porsi, model tetap akan menghitungnya sebagai satu porsi utuh dan model ini hanya bisa mendeteksi makanan saja. Pengguna dapat menambah gambar melalui kamera atau galeri. Implementasi fitur ini dapat dilihat pada Gambar 4.51.




Gambar 4. 51 Pencatatan Otomatis

Pada Gambar 4.52, dapat dilihat bahwa pada bagian kode ini, dapat dilihat bahwa bagian ini berfungsi untuk menangani pemilihan gambar dan proses analisis kandungan gula dalam makanan. Ketika pengguna menekan tombol galeri, sistem akan membuka pemilih gambar dan memungkinkan pengguna memilih gambar dari penyimpanan. Jika pengguna memilih untuk mengambil gambar menggunakan kamera, sistem akan terlebih dahulu memeriksa izin akses kamera sebelum membuka aplikasi kamera. Setelah gambar diperoleh, gambar tersebut akan ditampilkan dan diproses menggunakan fungsi *uCropUtils()* untuk pemotongan. Pada tahap analisis, gambar yang telah dipilih dikirim ke *scanViewModel.analyzeImage(),* yang kemudian mengamati hasil analisis berupa nama makanan dan jumlah kandungan gula. Setelah itu, sistem akan menghasilkan penjelasan ilmiah terkait makanan yang dianalisis melalui *scanViewModel.generateScientificExplanation().* Hasil analisis ini kemudian ditampilkan dalam sebuah bottom sheet yang berisi informasi makanan, kandungan gula, dan penjelasan ilmiah. Jika terjadi kesalahan dalam pemrosesan, sistem akan menampilkan pesan peringatan kepada pengguna menggunakan Toast.

```
    private fun setupListeners() {
        // Menangani klik untuk memilih gambar dari galeri
        binding.galleryButton.setOnClickListener {
            val intent = Intent(Intent.ACTION_GET_CONTENT)
            intent.type = "image/*"
            val chooser = Intent.createChooser(intent, null)
            launcherIntentGallery.launch(chooser)
        }

        // Menangani klik untuk membuka kamera dan memeriksa izin
        binding.cameraButton.setOnClickListener {
```

```
            if (checkCameraPermission()) {
                launchCamera()
            } else {
                requestCameraPermission()
            }
        }

        // Menangani klik untuk analisis gambar dan memproses data
        binding.analyzeButton.setOnClickListener {
            croppedImageUri?.let { uri ->
                val file = File(uri.path)
                showLoading(true)
                scanViewModel.analyzeImage(file)

                // Mengamati hasil analisis gambar dan menampilkan
penjelasan ilmiah
                scanViewModel.foodInfo.observe(this) { result ->
                    result?.let { (foodName, sugarContent) ->

scanViewModel.generateScientificExplanation(foodName,
sugarContent?.toString())


scanViewModel.scientificExplanation.observe(this) { explanation ->
                            val       myBottomSheetFragment       =
MyBottomSheetFragment.newInstance(
                                foodName,  sugarContent?.toString(),
explanation, uri
                            )

myBottomSheetFragment.show(supportFragmentManager,
myBottomSheetFragment.tag)
                            showLoading(false)
                        }
                        scanViewModel.foodInfo.removeObservers(this)
                    } ?: run {
                        showLoading(false)
                        Toast.makeText(this,    "Gagal    memproses
analisis!", Toast.LENGTH_SHORT).show()
                    }
                }
            }
        }
    }

    // Menangani hasil dari galeri untuk mendapatkan gambar yang
dipilih
    private val launcherIntentGallery = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
    ) { result ->
        if (result.resultCode == RESULT_OK) {
            val selectedImg = result.data?.data
            selectedImg?.let { uri ->
                initImageUri = uri
                showImage()
                uCropUtils(uri)
            } ?: showToast(getString(R.string.image_get_uri_error))
        }
    }
```

```
// Menangani hasil dari kamera untuk mendapatkan foto yang diambil
private val launcherIntentCamera = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult()
) { result ->
    if (result.resultCode == RESULT_OK) {
        initImageUri = photoUri
        showImage()
        uCropUtils(photoUri)
    } else {
        showToast(getString(R.string.image_capture_error))
    }
}
```

Gambar 4. 52 *Code* Pencatatan Otomatis

### 4.5.13 Halaman Hasil Pencatatan Otomatis

Halaman hasil Hitung Otomatis menampilkan nama makanan dan jumlah kandungan gula yang terdeteksi. Deskripsi ini dihasilkan menggunakan teknologi AI Gemini, yang memberikan informasi terkait makanan secara otomatis. Pengguna dapat menambah jumlah item dan mengonfirmasi melalui tombol "Tambah". Implementasi halaman hasil pindai dapat dilihat pada Gambar 4.53.




Gambar 4. 53 Hasil Pencatatan Otomatis

Pada Gambar 4.54, dapat dilihat bahwa pada bagian kode ini menangani tampilan hasil analisis makanan dalam bottom sheet serta memungkinkan pengguna menyimpan data yang telah diproses. Data seperti nama makanan, kandungan gula, porsi, dan informasi penjelasan ilmiah diambil dari *arguments* dan ditampilkan di antarmuka. Pengguna dapat menyesuaikan porsi makanan dengan tombol *Increment* dan *Decrement*, yang akan memperbarui jumlah total gula berdasarkan porsi yang dipilih. Saat tombol `*Save*` ditekan, sistem mengambil nama

makanan, jumlah gula per porsi, porsi, dan informasi penjelasan. Jika gambar makanan tersedia, sistem mengunggahnya bersama informasi terkait menggunakan *scanViewModel.uploadScanImage()*. Setelah pengunggahan selesai, bottom sheet ditutup, dan pengguna diarahkan kembali ke *ScannerActivity*.

```
    // Mengambil data dari arguments
    val foodName = arguments?.getString("foodName") ?: "Unknown"
    val sugarContent = arguments?.getString("sugarContent") ?: "0"
    val   explanation   =   arguments?.getString("explanation")   ?:   "No
explanation available"
    croppedImageUri = arguments?.getParcelable("imageUri")

    // Menampilkan data ke UI
    binding.foodName.text = foodName
    binding.sugarContent.text = "$sugarContent"
    binding.scientificOfExplanation.text = explanation

    // Listener untuk tombol Increment
    binding.btnIncrement.setOnClickListener {
        currentValue++
        binding.tvValue.text = currentValue.toString()
    }

    // Listener untuk tombol Decrement
    binding.btnDecrement.setOnClickListener {
        if (currentValue > 1) {
            currentValue-- // Kurangi currentValue
        }
        binding.tvValue.text = currentValue.toString()
    }

    // Listener untuk tombol Save:
    binding.saveButton.setOnClickListener {
        val foodName = binding.foodName.text.toString()
        val                          sugarAmount                                =
binding.sugarContent.text.toString().toDoubleOrNull() ?: 0.0
        val portion = binding.tvValue.text.toString().toDoubleOrNull() ?:
0.0
        val totalSugar = sugarAmount * portion

        croppedImageUri?.let { uri ->
            val imageFile = File(uri.path!!)
            val imageBody = MultipartBody.Part.createFormData(
                "gambar",                                  imageFile.name,
imageFile.asRequestBody("image/jpeg".toMediaTypeOrNull())
            )
            val                       foodNameRequest                         =
foodName.toRequestBody("text/plain".toMediaTypeOrNull())
            val                     sugarAmountRequest                        =
totalSugar.toString().toRequestBody("text/plain".toMediaTypeOrNull())
            val                         servingSize                           =
portion.toString().toRequestBody("text/plain".toMediaTypeOrNull())
            val                         inputMethod                           =
"scan".toRequestBody("text/plain".toMediaTypeOrNull())

            // Mengunggah data ke ViewModel
            scanViewModel.uploadScanImage(imageBody,      foodNameRequest,
sugarAmountRequest, servingSize, inputMethod)
```

```
            dismiss()
            startActivity(Intent(activity, ScannerActivity::class.java))
        }
    }
```

Gambar 4. 54 *Code* Hasil Pencatatan Otomatis

### 4.5.14 Halaman Chat Bot

Pada halaman Chat Bot, pengguna dapat berkomunikasi dengan AI Gemini untuk mendapatkan informasi seputar kesehatan dan konsumsi gula. Pengguna dapat menanyakan jumlah gula dalam makanan atau tips menjaga pola makan sehat. Namun, fitur ini terbatas hanya pada pertanyaan yang berkaitan dengan kesehatan dan konsumsi gula, dan tidak dapat digunakan untuk pertanyaan di luar topik tersebut. Implementasi halaman ini dapat dilihat pada Gambar 4.55.




Gambar 4. 55 Chat Bot

Pada Gambar 4.56, dapat dilihat bahwa pada bagian kode ini mengelola sistem pengiriman pesan dalam fitur chatbot yang berfokus pada konsumsi gula harian. Saat pengguna menekan tombol Send, sistem mengambil teks dari kolom input dan memeriksa apakah pesan tidak kosong. Jika ada teks yang dimasukkan, pesan akan dikirim ke *chatViewModel.sendMessage()*, lalu input dikosongkan. Sebaliknya, jika pesan kosong, sistem hanya mencatat log tanpa mengirim pesan.

Untuk menjalankan chatbot, kode ini menggunakan model generatif gemini-2.0-flash-exp dengan berbagai pengaturan, seperti temperature, topK, topP, serta batas maksimal output

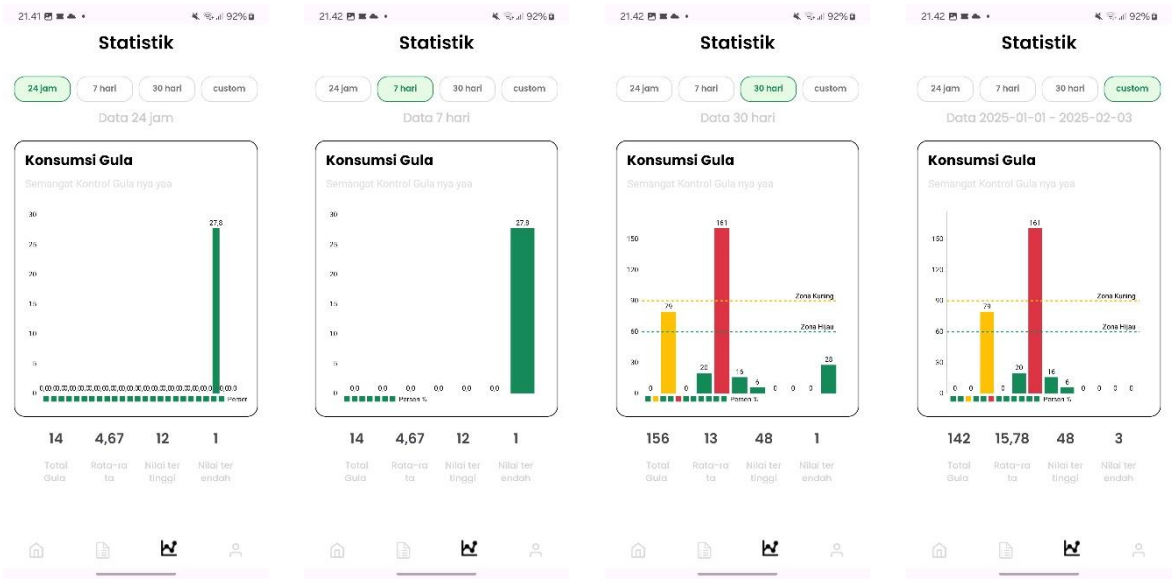
token. Model ini digunakan untuk memberikan respons yang relevan terhadap pertanyaan pengguna.

```
    // Tombol kirim
            binding.sendBtn.setOnClickListener {
                val message =
binding.messageEditText.text.toString()
                if (message.isNotEmpty()) {
                    chatViewModel.sendMessage(message)
                    binding.messageEditText.text.clear()
                } else {
                    Log.d("ChatActivity", "Message is empty")
                }
            }
    val model = GenerativeModel(
     modelName = "gemini-2.0-flash-exp",
     apiKey = Constants.apikey,
     generationConfig = generationConfig {
         temperature = 1f
         topK = 64
         topP = 0.95f
         maxOutputTokens = 800
         responseMimeType = "text/plain"
     }
)
```

Gambar 4. 56 *Code* Halaman Chat Bot

### 4.5.15 Halaman Statistik

Halaman Statistik menampilkan informasi terkait konsumsi gula dalam berbagai periode waktu, yaitu 24 jam, 7 hari, 30 hari, atau custom. Setiap grafik menunjukkan jumlah total konsumsi gula, dengan indikator warna untuk memudahkan pemahaman: warna hijau menunjukkan konsumsi gula dalam batas yang sehat, kuning menunjukkan konsumsi gula yang perlu perhatian, dan merah menunjukkan konsumsi gula yang berlebihan. Pada setiap periode, grafik juga mencakup data seperti rata-rata konsumsi gula, nilai tertinggi, dan nilai terendah. Di bawah grafik, terdapat informasi tambahan tentang total konsumsi gula dalam periode tersebut. Halaman statistik ini membantu pengguna untuk melacak dan memonitor konsumsi gula mereka secara lebih jelas dan terstruktur. Implementasi halaman statistik dapat dilihat pada Gambar 4.57.




Gambar 4. 57 Statistik

Pada Gambar 4.58, dapat dilihat bahwa pada bagian kode ini mengatur tombol-tombol yang digunakan untuk memilih rentang waktu dalam tampilan statistik konsumsi gula. Terdapat empat tombol utama, yaitu "24 jam", "7 hari", "30 hari", dan "custom", yang dikelompokkan dalam daftar untuk mempermudah pengelolaan event klik. Setiap.

Ketika salah satu tombol ditekan, sistem akan mengembalikan semua tombol ke tampilan default terlebih dahulu, kemudian mengubah tampilan tombol yang ditekan menjadi tampilan terpilih. Jika tombol yang dipilih adalah "custom", maka sistem akan menampilkan pemilih rentang tanggal menggunakan fungsi *showDateRangePicker()*. Sebaliknya, jika tombol yang dipilih adalah salah satu dari rentang waktu yang telah ditentukan, maka teks pada bagian dataTitle akan diperbarui sesuai dengan pilihan pengguna, dan sistem akan memanggil fungsi *getStatistics(title)* pada statistikViewModel untuk mengambil data statistik berdasarkan rentang waktu yang dipilih.

```
    private fun setupButtonListeners() {
            val buttons = listOf(
                Pair(binding.btn24jam, "24 jam"),
                Pair(binding.btn7hari, "7 hari"),
                Pair(binding.btn30Hari, "30 hari"),
                Pair(binding.btnCustom, "custom")
            )

            val defaultBackground = R.drawable.rectangle_outline
            val selectedBackground = R.drawable.rectangle_filled
            val defaultTextColor = R.color.darkGrey
            val selectedTextColor = R.color.txstatistik

            buttons.forEach { (button, title) ->
                button.setOnClickListener {
                    buttons.forEach { (btn, _) ->
                        btn.setBackgroundResource(defaultBackground)
```

```
                        val textView = btn.getChildAt(0) as?
TextView

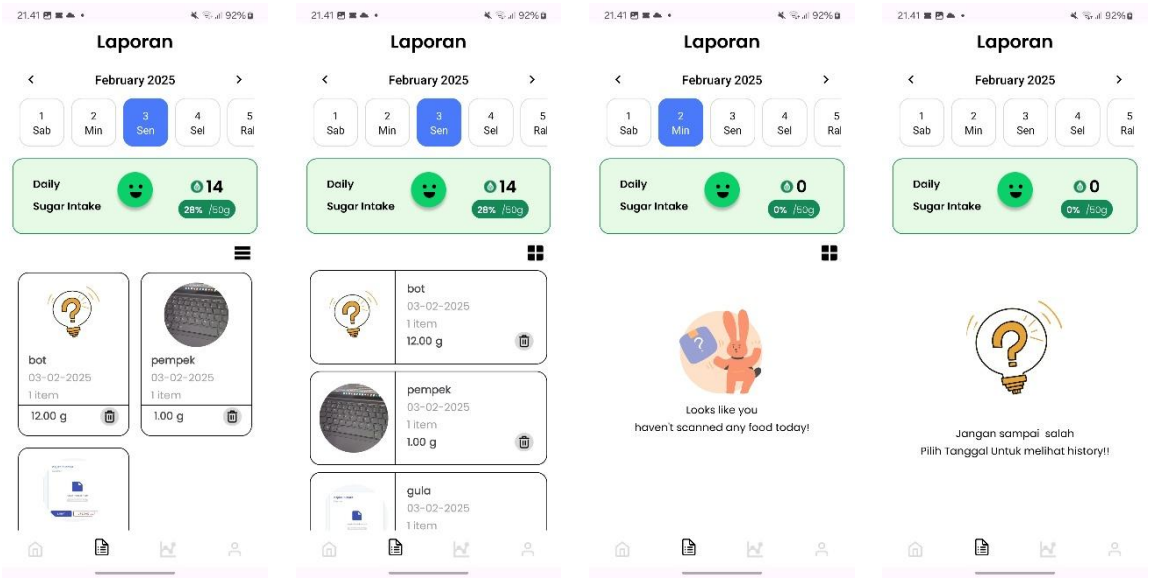
textView?.setTextColor(resources.getColor(defaultTextColor, null))
                    }
                    button.setBackgroundResource(selectedBackground)
                    val selectedTextView = button.getChildAt(0) as?
TextView

selectedTextView?.setTextColor(resources.getColor(selectedTextColor,
null))
                    if (title == "custom") {
                        showDateRangePicker()
                    } else {
                        binding.dataTitle.text = "Data $title"
                        statistikViewModel.getStatistics(title)
                    }
                }
            }
        }
```

Gambar 4. 58 *Code* Halaman Statistik

### 4.5.16 Halaman Laporan

Halaman Laporan menampilkan konsumsi gula harian pengguna dalam bentuk grafik dan persentase, serta daftar makanan yang telah dipindai pada hari yang dipilih. Setiap makanan ditampilkan beserta kandungan gula, membantu pengguna untuk mengevaluasi kebiasaan makan mereka dan meningkatkan kesadaran nutrisi. Dengan demikian, pengguna dapat memilih makanan secara bijak untuk menghindari konsumsi gula yang berlebihan. Halaman ini juga memungkinkan pengguna untuk melihat laporan berdasarkan tanggal tertentu. Implementasi halaman laporan dapat dilihat pada Gambar 4.59.



Gambar 4. 59 Laporan Riwayat Harian

Pada Gambar 4.60, dapat dilihat bahwa pada bagian kode ini menangani tampilan riwayat konsumsi gula berdasarkan tanggal yang dipilih pengguna. Saat tanggal dipilih, sistem mengonversi formatnya ke ISO dan mengambil data konsumsi gula melalui *historyViewModel.fetchConsumptionHistory()*. Jika data kosong, UI diperbarui dengan pesan kosong; jika ada, total konsumsi dihitung dan ditampilkan.

Fungsi *updateUI* memperbarui tampilan berdasarkan total konsumsi dan batas harian. Jika konsumsi melebihi 90%, UI berubah merah sebagai peringatan, 60-90% menjadi kuning, dan di bawah 60% tetap hijau. Dengan cara ini, pengguna mendapat gambaran jelas tentang pola konsumsi gulanya.

```
    override fun onItemClick(ddMmYy: String, dd: String, day: String) {
            val isoDate = convertDateToIsoFormat(ddMmYy)
            historyViewModel.fetchConsumptionHistory(isoDate)


historyViewModel.consumptionHistory.observe(viewLifecycleOwner) { history
->
                if (history.isNullOrEmpty()) {
                    val totalSugar =
historyViewModel.calculateTotalSugar(history)
                    val dailyLimit
=historyViewModel.resultResponse.value?.maksimalGula?.toDoubleOrNull()?.t
oInt()?: 0
                    updateUI(totalSugar, dailyLimit)
                    showEmptyView()
                } else {
                    val totalSugar =
historyViewModel.calculateTotalSugar(history)
                    val dailyLimit =
historyViewModel.resultResponse.value?.maksimalGula?.toDoubleOrNull()?.to
Int()?: 0
                    updateUI(totalSugar, dailyLimit)
                    hideEmptyView()
                }
            }
            historyViewModel.userResponse.observe(viewLifecycleOwner) {
response ->
                response?.let {
                    historyViewModel.fetchConsumptionHistory(isoDate)
                }
            }
        }

        private fun updateUI(totalSugar: Double, dailyLimit: Int? = null)
{
            binding.dailySugarIntakeCard.tvSugarIntakeValue.text =
totalSugar.toInt().toString()
            dailyLimit?.let {
                val percentageConsumed = calculatePercentage(totalSugar,
it)
                binding.dailySugarIntakeCard.tvSugarIntakePercentage.text
= "$percentageConsumed%"
```

```
                binding.dailySugarIntakeCard.tvSugarIntakeMax.text =
"/${it}g"
                when {
                    percentageConsumed > 90 -> {

binding.dailySugarIntakeCard.ivSugarIntake.setImageResource(R.drawable.ic
_angry_small)

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.rectan
gle_filled_red)

binding.dailySugarIntakeCard.ivBloodSugar.setImageResource(R.drawable.ic_
blood_sugar)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawable.sha
pe_red)
                    }
                    percentageConsumed > 60 ->{

binding.dailySugarIntakeCard.ivSugarIntake.setImageResource(R.drawable.ic
_good_small)

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.rectan
gle_filled_yellow)

binding.dailySugarIntakeCard.ivBloodSugar.setImageResource(R.drawable.ic_
blood_yelow)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawable.sha
pe_yelow)

binding.dailySugarIntakeCard.tvSugarIntakePercentage.setTextColor(Context
Compat.getColor(requireContext(), R.color.black))

binding.dailySugarIntakeCard.tvSugarIntakeMax.setTextColor(ContextCompat.
getColor(requireContext(), R.color.black))
                    }
                    else -> {

binding.dailySugarIntakeCard.ivSugarIntake.setImageResource(R.drawable.ic
_happy_small)

binding.dailySugarIntakeCard.card.setBackgroundResource(R.drawable.rectan
gle_filled)

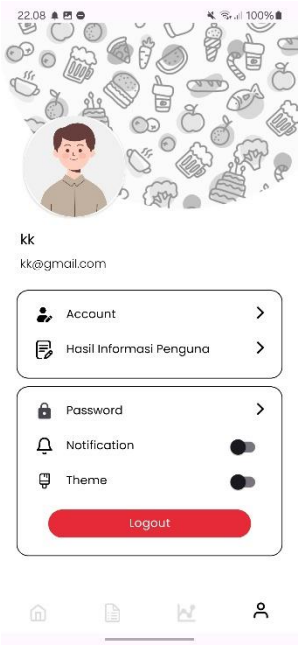
binding.dailySugarIntakeCard.ivBloodSugar.setImageResource(R.drawable.ic_
blood_green)

binding.dailySugarIntakeCard.percent.setBackgroundResource(R.drawable.sha
pe_green)
                    }
                }
            }
        }
```

Gambar 4. 60 *Code* Halaman Laporan Riwayat Harian

### 4.5.17 Halaman Profile

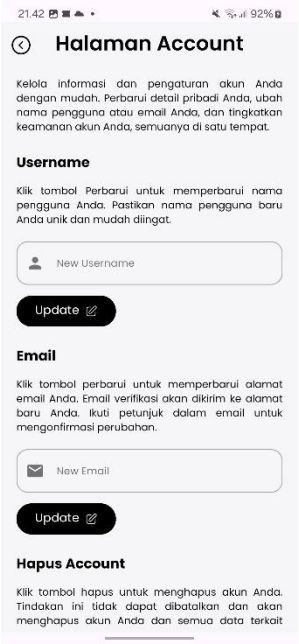
Halaman Profil memungkinkan pengguna untuk melihat dan mengelola informasi akun mereka, termasuk nama, email, dan pengaturan akun lainnya. Pengguna dapat mengakses pengaturan untuk mengubah password, mengaktifkan atau menonaktifkan notifikasi, dan mengganti tema aplikasi. Terdapat juga opsi untuk melihat hasil informasi pengguna terkait konsumsi gula dan aktivitas lainnya. Untuk keluar dari akun, pengguna dapat memilih tombol "Logout". Halaman ini memberikan kontrol penuh atas pengaturan pribadi dalam aplikasi. Implementasi halaman profil dapat dilihat pada Gambar 4.60.




Gambar 4. 61 Profile

### 4.5.18 Halaman Account

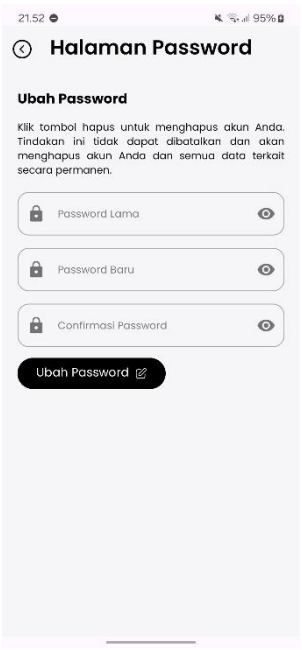
Halaman Account memungkinkan pengguna untuk mengelola dan memperbarui informasi pribadi mereka, seperti nama pengguna dan email. Pengguna dapat mengklik tombol "*Update*" untuk memperbarui nama pengguna atau alamat email mereka. Jika mengubah email, pengguna akan menerima email verifikasi untuk mengonfirmasi perubahan. Halaman ini juga menyediakan opsi untuk menghapus akun, yang akan menghapus semua data terkait akun secara permanen. Implementasi halaman account dapat dilihat pada Gambar 4.61.




Gambar 4. 62 Account

### 4.5.19 Halaman Reset Password

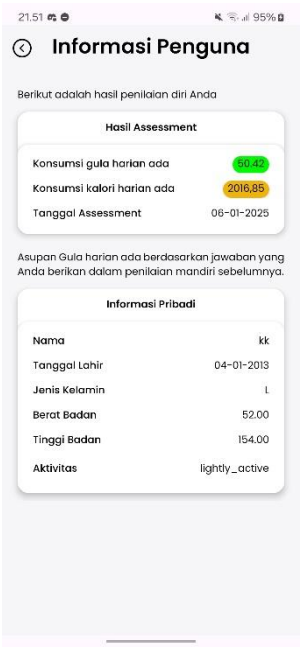
Halaman Password memungkinkan pengguna untuk mengganti password mereka. Pengguna dapat memasukkan password lama, kemudian password baru, dan mengonfirmasi password baru tersebut. Setelah perubahan dilakukan, pengguna dapat mengklik tombol "Ubah Password" untuk memperbarui password akun mereka. Implementasi halaman password dapat dilihat pada Gambar 4.63.




Gambar 4. 63 Password

### 4.5.20 Halaman Hasil Informasi Penguna

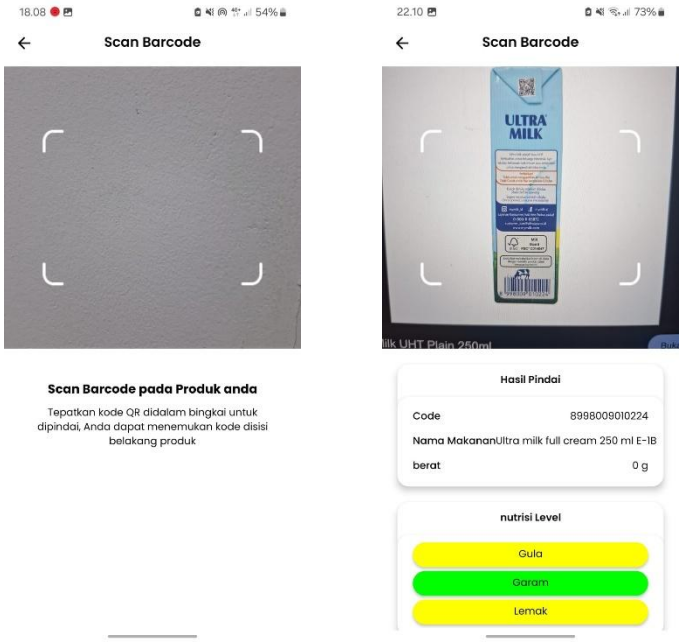
Halaman Informasi Pengguna menampilkan hasil perhitungan konsumsi gula harian dan kalori berdasarkan BMR (Basal Metabolic Rate). Maksimum konsumsi gula dihitung dari 10% total kalori harian, yang kemudian dibagi 4. Halaman ini juga mencakup informasi pribadi pengguna, seperti nama, tanggal lahir, berat badan, tinggi badan, dan aktivitas, yang digunakan untuk menyesuaikan rekomendasi konsumsi gula. Implementasi halaman ini dapat dilihat pada Gambar 4.64.




Gambar 4. 64 Hasil Informasi Penguna

### 4.5.21 Halaman Bar*Code*

Halaman be*Code* merupakan halaman yang mana penguna bisa melakukan scan bar*Code* makana atau minuman dan pengguna mendapatkan informasi berupa nama, berat, nutrisi level, dan kandungan nutrisi pada peroduk makanan atau minuman yang di scan bar*Code*. Implementasi ini dapat di lihat pada gambar 4.65.




Gambar 4. 65 Bar*Code* Scan

Pada Gambar 4.66, dapat dilihat bahwa pada bagian kode ini proses pemindaian bar*Code* dilakukan menggunakan CameraX ImageAnalysis, yang kemudian digunakan untuk mengambil informasi produk dari Open Food Facts API. Fungsi utama yang bertanggung jawab dalam proses ini adalah *analysisUseCase*, yang dikonfigurasi dengan strategi STRATEGY_KEEP_ONLY_LATEST untuk memastikan hanya gambar terbaru yang diproses. Gambar yang diambil dikonversi ke format InputImage dan dipindai menggunakan bar*Code*Scanner. Jika bar*Code* berhasil dikenali, nilainya akan dikirim ke fungsi *getProductInfo()* untuk mendapatkan informasi produk.

Fungsi *getProductInfo()* menggunakan Retrofit untuk melakukan permintaan ke Open Food Facts API berdasarkan bar*Code* yang dipindai. Jika permintaan berhasil, UI diperbarui dengan informasi produk, seperti nama, berat, kandungan gula, lemak, protein, serat, dan garam. Selain itu, warna kartu indikator nutrisi akan diperbarui berdasarkan tingkat kandungan gizi produk tersebut. Jika produk tidak ditemukan atau terjadi kegagalan dalam mengambil data, UI akan menyembunyikan informasi produk dan menampilkan pesan kesalahan kepada pengguna.

```
    val analysisUseCase = ImageAnalysis.Builder()

.setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
    .build()
    .also {
        it.setAnalyzer(Executors.newSingleThreadExecutor()) {
imageProxy ->
            val mediaImage = imageProxy.image
            if (mediaImage != null) {
```

```
                val image = InputImage.fromMediaImage(mediaImage,
imageProxy.imageInfo.rotationDegrees)
                barCodeScanner.process(image)
                    .addOnSuccessListener { barCodes ->
                        for (barCode in barCodes) {
                            val barCodeValue = barCode.rawValue
                            if (barCodeValue != null) {
                                getProductInfo(barCodeValue)
                            }
                        }
                    }
                    .addOnFailureListener {
                        Log.e("BarCode", "Gagal memindai")
                    }
                    .addOnCompleteListener {
                        imageProxy.close()
                    }
            }
        }
    }

   fun getProductInfo(barCode: String) {
    val retrofit = Retrofit.Builder()
        .baseUrl("https://world.openfoodfacts.org/")
        .addConverterFactory(GsonConverterFactory.create())
        .build()

    val api = retrofit.create(ApiService::class.java)
    val call = api.getProduct(barCode)
    call.enqueue(object : Callback<ProductResponse> {
        override fun onResponse(call: Call<ProductResponse>,
response: Response<ProductResponse>) {
            if (response.isSuccessful) {
                val product = response.body()?.product
                val Code = response.body()?.Code
                binding.Code.text = Code
                if (product != null) {
                    runOnUiThread {
                        binding.assessmentResultCardView.visibility
= View.VISIBLE

binding.personalInformationCardView.visibility = View.VISIBLE
                        binding.nutrisicardview.visibility =
View.VISIBLE
                        binding.title.visibility =View.GONE

                        binding.tvNameMakanan.text =
product.productName ?: "Tidak ada nama"
                        binding.berat.text =
"${product.productQuantity ?: 0} g"
                        binding.GulaMakanan.text =
"${product.nutriments?.sugars100g ?: 0.0} g / 100g"
                        binding.lemak.text =
"${product.nutriments?.fat100g ?: 0.0} g / 100 g"
                        binding.protein.text =
"${product.nutriments?.proteins100g ?: 0.0} g / 100g"
                        binding.Serat.text =
"${product.nutriments?.fiber100g ?: 0.0} g / 100g"
                        binding.garam.text =
"${product.nutriments?.salt100g ?: 0.0} g / 100g"
```

```
                        updateCardColor(binding.cardSugar,
product.nutrientLevels?.sugars)
                        updateCardColor(binding.cardFat,
product.nutrientLevels?.fat)
                        updateCardColor(binding.cardSalt,
product.nutrientLevels?.salt)
                    }
                } else {

                    binding.assessmentResultCardView.visibility =
View.GONE
                    binding.personalInformationCardView.visibility
= View.GONE
                    binding.nutrisicardview.visibility = View.GONE
                    binding.title.visibility =View.VISIBLE

                    Toast.makeText(this@LabelActivity, "Produk
tidak ditemukan", Toast.LENGTH_SHORT).show()
                }
            } else {
                // Sembunyikan semua CardView terlebih dahulu
                binding.assessmentResultCardView.visibility =
View.GONE
                binding.personalInformationCardView.visibility =
View.GONE
                binding.nutrisicardview.visibility = View.GONE
                binding.title.visibility =View.VISIBLE

                Toast.makeText(this@LabelActivity, "Gagal mengambil
data produk", Toast.LENGTH_SHORT).show()
            }
        }
        override fun onFailure(call: Call<ProductResponse>, t:
Throwable) {
            Log.e("API Error", "Gagal mengambil data produk:
${t.message}")
        }
    })
}
```

Gambar 4. 66 *Code* Halaman Bar*Code* Scan

### 4.5.22 Implementasi API

Tabel 4.5 merupakan daftar API yang diimplementasikan pada aplikasi Pemantauan Konsumsi Gula Harian. API ini digunakan untuk mengambil dan mengelola data profil pengguna, konsumsi gula, statistik konsumsi, dan artikel.

Tabel 4. 5 Implementasi API

| No. | Tabel | Metode | Fungsi | Endpoint |
|---|---|---|---|---|
| 1 | *Auth* | POST | *Register* | /api/auth/register |
| | | POST | *Login* | /api/auth/login |

| | | | | |
|---|---|---|---|---|
| | | PUT | *Update Email* | /api/auth/update/email |
| | | PUT | *Update Password* | /api/auth/update/password |
| | | PUT | *Update Username* | /api/auth/update/username |
| | | DELETE | *Delete Account* | /api/auth/delete |
| | | POST | *Logout* | /api/auth/logout |
| 2 | *User Profile* | POST | *Create Profile* | /api/informasi/create |
| | | GET | *Get Profile Status* | /api/informasi/profile/status |
| | | PUT | *Update Sugar* | /api/informasi/update/gula |
| 3 | *Consumption* | POST | *Upload Consumption* | /api/consumption/upload |
| | | GET | *Get Consumption History by Date* | /api/consumption/history |
| | | GET | *Get Total Sugar Today* | /api/consumption/total-sugar-today |
| | | DELETE | *Delete Consumption by ID* | /api/consumption/{recordId} |
| | | POST | *Analyze Image* | /api/consumption/analyze |
| 4 | *Statistics* | GET | *Get Consumption Statistics* | /api/statistik/statistics |
| 5 | *Articles* | GET | *Get Articles by Category* | /api/articles/category/{categoryId} |

## 4.6 Hasil Pegujian

### 4.6.1 Hasil Black Box Testing

Pengujian fungsional dilakukan menggunakan metode black box pada aplikasi Pemantauan Konsumsi Gula Harian. Pengujian ini berfokus pada keluaran yang diberikan berdasarkan skenario yang telah dibuat. Pengujian ini juga memastikan bahwa aplikasi Pemantauan Konsumsi Gula Harian dapat menangani kesalahan masukan dari pengguna dan

memberikan respons yang sesuai. Pengujian ini dilakukan secara langsung oleh peneliti untuk memastikan setiap fungsi berjalan sesuai dengan skenario.

Tabel 4. 6 Hasil Black Box Testing

| No. | Kasus Yang Diuji | Skenario Uji | Hasil yang Diharapkan | Hasil Pengujian |
|---|---|---|---|---|
| 1. | Halaman Login | Memasukkan email dan password yang benar | Pengguna berhasil login dan diarahkan ke halaman home pengguna | Berhasil |
| | | Memasukkan email dan password dengan salah password | Pengguna gagal login dan aplikasi memunculkan alert "Login gagal" | Berhasil |
| | | Memasukkan email dan password dengan salah email | Pengguna gagal login dan aplikasi memunculkan alert "Login gagal" | Berhasil |
| | | Mengosongkan salah satu isian field | Pengguna gagal login dan aplikasi memunculkan alert "Email dan password tidak boleh kosong" | Berhasil |
| 2. | Halaman Daftar | Memasukkan email, username, dan password | Pengguna berhasil daftar dan diarahkan ke halaman login | Berhasil |
| | | Mengosongkan salah satu isian field | Pengguna gagal daftar dan aplikasi memunculkan alert "Username, email, dan password tidak boleh kosong" | Berhasil |
| 3. | Halaman Create Profile | Memasukkan data profil lengkap (username, Tanggal lahir , etc.) | Pengguna berhasil membuat profil dan data profil disimpan | Berhasil |
| 4. | Halaman Home | Memilih halaman yang tersedia | Menampilkan halaman yang dipilih | Berhasil |
| | | Melihat informasi profil pengguna | Menampilkan informasi dari pengguna | Berhasil |
| | | Menghitung total gula konsumsi hari ini | Menampilkan total konsumsi gula hari ini | Berhasil |
| | | Mengaktifkan notifikasi untuk mengingatkan pengguna mencatat konsumsi gula pada waktu yang ditentukan dan pemeberitahuna | Pengguna menerima notifikasi pada waktu yang tepat dan sesuai dengan pengaturan yang telah | Berhasil |

| | | | | |
|---|---|---|---|---|
| | | Ketika konsumsi gula mendekati atau melebihi batas yang di saran kan | ditentukan, baik untuk mencatat konsumsi gula atau peringatan konsumsi gula berlebih. | |
| 7. | Halaman Statistik | Menampilkan statistik konsumsi gula berdasarkan periode | Menampilkan statistik konsumsi gula berdasarkan periode | Berhasil |
| 8. | Halaman Konsumsi | Mengupload konsumsi baru dengan gambar | Pengguna berhasil mengupload konsumsi baru | Berhasil |
| | | Menampilkan riwayat konsumsi berdasarkan tanggal | Menampilkan riwayat konsumsi yang sudah tercatat berdasarkan tanggal | Berhasil |
| | | Menghapus data konsumsi berdasarkan ID | Data konsumsi berhasil dihapus berdasarkan ID | Berhasil |
| | | Menganalisis gambar konsumsi | Aplikasi berhasil menganalisis gambar konsumsi | Berhasil |
| | | Menginput data gula secara manual | Pengguna berhasil menambahkan data konsumsi gula secara manual, dan data tampil di riwayat konsumsi gula. | Berhasil |
| 9. | Halaman Profil | Menampilkan username, email | Menampilkan username dan email | Berhasil |
| | | Mengedit username, email, password, dan delate acount | Data pengguna berhasil tersimpan dan informasi profil berubah | Berhasil |
| 10. | Halaman Artikel | Menampilkan artikel berdasarkan kategori | Menampilkan artikel sesuai kategori | Berhasil |
| 11 | Halaman Chatbot | Menggunakan chatbot untuk bertanya tentang pola makan sehat | Chatbot merespon dengan informasi yang relevan tentang pola makan sehat atau konsumsi gula | Berhasil |
| 12 | Halaman Infromsai Peguna | Melihat informasi terkait jumlah kalori dan gula yang dikonsumsi berdasarkan data diri yang dimasukkan oleh pengguna | Pengguna melihat total kalori dan maksimal jumlah gula yang di konsumsi | Berhasil |

| | | | | |
|---|---|---|---|---|
| 13 | Halaman barcode | Melakukan pemindaian barcode produk makanan/minuman dengan kamera | Pengguna melihat hasil pindai berupa nama makanan, berat, level nutrisi, dan nutrisi dari produk. | Berhasil |
| | | Memindai barcode produk yang tidak dikenal | Aplikasi menampilkan Toast "Produk tidak ditemukan" | Berhasil |

### 4.6.2 Hasil System Usability Scale (SUS)

Pengujian usabilitas dilakukan dengan menggunakan System Usability Scale (SUS) kepada 14 pengguna, yang mencakup remaja dan dewasa muda, yaitu individu berusia 13–35 tahun. Pengujian dimulai dengan pengguna mencoba aplikasi Pemantauan Konsumsi Gula Harian atau Glucofit. Setelah mencoba aplikasi, mereka diminta untuk mengisi kuesioner yang terdiri dari 10 pertanyaan SUS yang ditunjukkan pada Tabel 3.2. Data yang diperoleh kemudian disimpan dalam tabel yang dapat dilihat pada Tabel 4.7.

Tabel 4. 7 Tabel Hasil Kuesioner Pengujian SUS

| Responden | Pertanyaan | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | K1 | K2 | K3 | K4 | K5 | K6 | K7 | K8 | K9 | K10 |
| R1 | 5 | 2 | 5 | 2 | 4 | 2 | 5 | 1 | 4 | 2 |
| R2 | 3 | 2 | 3 | 2 | 4 | 1 | 4 | 1 | 4 | 2 |
| R3 | 2 | 4 | 5 | 3 | 5 | 1 | 5 | 2 | 4 | 2 |
| R4 | 4 | 1 | 5 | 2 | 5 | 1 | 4 | 2 | 5 | 2 |
| R5 | 5 | 2 | 4 | 3 | 4 | 2 | 4 | 2 | 2 | 3 |
| R6 | 5 | 1 | 5 | 1 | 4 | 4 | 4 | 1 | 4 | 2 |
| R7 | 5 | 2 | 5 | 2 | 5 | 1 | 5 | 1 | 5 | 4 |
| R8 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 |
| R9 | 5 | 1 | 5 | 1 | 5 | 1 | 2 | 1 | 5 | 2 |
| R10 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 |
| R11 | 5 | 1 | 5 | 2 | 5 | 1 | 5 | 1 | 4 | 1 |
| R12 | 4 | 2 | 5 | 2 | 5 | 1 | 4 | 1 | 4 | 2 |
| R13 | 5 | 1 | 5 | 1 | 4 | 1 | 5 | 5 | 5 | 1 |
| R14 | 4 | 2 | 3 | 1 | 3 | 2 | 4 | 1 | 3 | 2 |

Berdasarkan data yang telah diperoleh sebelumnya, dilakukan perhitungan menggunakan rumus yang telah dijelaskan pada Bab III untuk mendapatkan hasil akhir dari pengujian. Proses perhitungan ini bertujuan untuk mengevaluasi tingkat kegunaan aplikasi secara lebih mendalam. Hasil dari perhitungan tersebut disajikan dalam Tabel 4.8, yang memberikan gambaran yang jelas mengenai skor dan analisis terkait usabilitas aplikasi

Tabel 4. 8 Tabel Hasil Perhitungan SUS

| Responden | Pertanyaan | | | | | | | | | | Skor | Skor x 2.5 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | K1 | K2 | K3 | K4 | K5 | K6 | K7 | K8 | K9 | K10 | | |
| R1 | 5 | 2 | 5 | 2 | 4 | 2 | 5 | 1 | 4 | 2 | 34 | 85 |
| R2 | 3 | 2 | 3 | 2 | 4 | 1 | 4 | 1 | 4 | 2 | 30 | 75 |
| R3 | 2 | 4 | 5 | 3 | 5 | 1 | 5 | 2 | 4 | 2 | 29 | 72,5 |
| R4 | 4 | 1 | 5 | 2 | 5 | 1 | 4 | 2 | 5 | 2 | 35 | 87,5 |
| R5 | 5 | 2 | 4 | 3 | 4 | 2 | 4 | 2 | 2 | 3 | 27 | 67,5 |
| R6 | 5 | 1 | 5 | 1 | 4 | 4 | 4 | 1 | 4 | 2 | 33 | 82,5 |
| R7 | 5 | 2 | 5 | 2 | 5 | 1 | 5 | 1 | 5 | 4 | 35 | 87,5 |
| R8 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 5 | 1 | 40 | 100 |
| R9 | 5 | 1 | 5 | 1 | 5 | 1 | 2 | 1 | 5 | 2 | 36 | 90 |
| R10 | 3 | 3 | 3 | 3 | 3 | 3 | 4 | 3 | 3 | 3 | 21 | 52,5 |
| R11 | 5 | 1 | 5 | 2 | 5 | 1 | 5 | 1 | 4 | 1 | 38 | 95 |
| R12 | 4 | 2 | 5 | 2 | 5 | 1 | 4 | 1 | 4 | 2 | 34 | 85 |
| R13 | 5 | 1 | 5 | 1 | 4 | 1 | 5 | 5 | 5 | 1 | 35 | 87,5 |
| R14 | 4 | 2 | 3 | 1 | 3 | 2 | 4 | 1 | 3 | 2 | 29 | 72,5 |
| Rata-Rata | | | | | | | | | | | | 81,42857 |

Berdasarkan hasil pengujian System Usability Scale (SUS) yang dilakukan kepada 14 pengguna, dapat disimpulkan bahwa aplikasi Pemantauan Konsumsi Gula Harian (Glucofit) mendapatkan rata-rata skor 81,43, yang masuk ke dalam kategori grade "B", dengan penilaian adjective rating "*Excellent*" dan acceptability range "*Acceptable*".

Hasil perhitungan ini menunjukkan bahwa aplikasi Glucofit mudah digunakan oleh pengguna, dengan tingkat usability yang sangat baik. Sebagian besar responden memberikan skor yang tinggi, mengindikasikan bahwa aplikasi ini memiliki antarmuka yang intuitif, efisien, dan memberikan pengalaman yang memuaskan.

Untuk memberikan gambaran yang lebih mendalam mengenai tingkat usability aplikasi, perhitungan terhadap lima variabel usability dilakukan menggunakan data yang diperoleh melalui pengujian SUS. Berikut adalah hasil perhitungan rata-rata untuk masing-masing variabel usability yang disajikan dalam tabel berikut:

Tabel 4. 9 Tabel Hasil lima Variabel Usability

| **Variabel Usability** | **List Item Pertayaan** | **Rata – rata Aspek Usablity** | **Interpretasi** |
|---|---|---|---|
| *Learnability* | K1, K3, K7 | 4.36 | Skor tinggi menunjukkan bahwa pengguna merasa aplikasi sangat |

| | | | |
|---|---|---|---|
| | | | mudah dipelajari dan dapat digunakan dengan cepat. |
| *Efficiency* | K5, K9 | 4.21 | Menunjukkan bahwa pengguna merasa aplikasi efisien dan dapat menyelesaikan tugas dengan cepat dan mudah. |
| *Errors* | K2, K4, K6,K8 | 1.67 | Skor rendah menandakan bahwa pengguna jarang mengalami kesalahan atau kesulitan saat menggunakan aplikasi. |
| *Satisfaction* | K1, K5, K9 | 4.18 | Pengguna merasa puas dengan pengalaman menggunakan aplikasi, menunjukkan tingkat kepuasan yang tinggi. |

Berdasarkan hasil analisis perhitungan System Usability Scale (SUS), dapat disimpulkan bahwa aplikasi Glucofit memiliki tingkat kegunaan yang secara keseluruhan cukup baik, dengan rata-rata skor 81,43%. Meskipun demikian, terdapat beberapa area yang perlu diperbaiki untuk meningkatkan kepuasan pengguna.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil dari implementasi pengembangan dan pengujian aplikasi pemantauan konsumsi gula harian yang telah dilakukan pada bab sebelumnya, dapat ditarik kesimpulan sebagai berikut:

a. Pengembangan aplikasi untuk pemantauan konsumsi gula harian dilakukan dengan menggunakan metode pengembangan Agile. Proses ini dilakukan secara iteratif, dimulai dengan analisis kebutuhan pengguna, dilanjutkan dengan perancangan aplikasi dan pengembangan fitur, serta pengujian berkala. Setiap tahapan dilakukan dengan evaluasi dan perbaikan secara berkelanjutan untuk memastikan aplikasi sesuai dengan harapan pengguna. Dengan pendekatan ini, aplikasi dapat terus berkembang dan disesuaikan dengan umpan balik yang diperoleh selama setiap sprint.
b. Hasil pengujian pada aplikasi menggunakan metode black box testing menunjukkan bahwa aplikasi berjalan sesuai dengan skenario yang diinginkan. Selain itu, berdasarkan hasil pengujian usability menggunakan System Usability Scale (SUS), aplikasi ini memperoleh skor rata-rata akhir 81,43, yang menunjukkan bahwa aplikasi dapat digunakan dengan mudah oleh pengguna. Skor tersebut termasuk dalam kategori grade "B", dengan penilaian adjective rating "*Excellent"* dan acceptability range dalam nilai "*Acceptable*", yang menandakan bahwa aplikasi ini memiliki tingkat kegunaan yang sangat baik.

## 5.2 Saran

Aplikasi GlucoFit untuk pemantauan konsumsi gula harian berbasis Android yang telah dikembangkan masih memiliki beberapa kekurangan yang dapat ditingkatkan serta peluang pengembangan lebih lanjut. Adapun saran yang dapat diberikan oleh peneliti untuk pengembangan berikutnya adalah sebagai berikut:

a. Disarankan agar tampilan antarmuka aplikasi terus diperbarui mengikuti perkembangan tren desain terkini dalam bidang kesehatan digital dan teknologi informasi. Pembaruan antarmuka secara berkala dapat membuat aplikasi tetap relevan dan menarik, baik bagi pengguna baru maupun pengguna lama. Tampilan yang intuitif dan modern akan meningkatkan kenyamanan serta pengalaman pengguna secara keseluruhan.
b. Penyempurnaan aspek gamifikasi ringan seperti penambahan tantangan harian atau pencapaian pribadi yang tidak bersifat kompetitif dapat menjadi strategi efektif untuk

meningkatkan keterlibatan dan motivasi pengguna. Dengan pendekatan yang menyenangkan, pengguna didorong untuk terus menjaga kebiasaan sehat tanpa merasa terbebani.

c. Pengembangan fitur deteksi makanan berbasis kecerdasan buatan perlu ditingkatkan, agar tidak hanya mengenali makanan tetapi juga mampu menghitung kandungan gula berdasarkan porsi konsumsi yang terdeteksi baik itu makahan maupun minuman. Selain itu, fitur ini sebaiknya diperluas untuk memberikan informasi nutrisi yang lebih lengkap, seperti jumlah kalori, karbohidrat, dan zat gizi lainnya, sehingga pengguna dapat memantau asupan harian mereka secara lebih menyeluruh dan akurat.

d. Aplikasi ini juga memiliki potensi untuk dikembangkan lebih luas, tidak hanya sebatas pemantauan konsumsi gula, melainkan menjadi platform pemantauan pola makan secara menyeluruh. termasuk memantau asupan kalori, karbohidrat, dan zat gizi lainnya, aplikasi dapat menjadi solusi digital yang lebih komprehensif dalam mendukung gaya hidup sehat.

# DAFTAR PUSTAKA


Abrue, M. (2022, April 23). *Kalkulator Persamaan Harris-Benedict untuk Ahli Gizi*. Nutrium. https://nutrium.com/blog/harris-benedict-equation-calculator-for-nutrition-professionals/

Andalas, S. F. T. (2024, June 12). *209,3 Juta Orang di Indonesia Menggunakan Smartphone pada Tahun 2023*. GoodStats. https://data.goodstats.id/statistic/2093-juta-orang-di-indonesia-menggunakan-smartphone-pada-tahun-2023-cbha0

Anupam Chugh. (2022, August 4). *Retrofit Android Example Tutorial*. Https://Www.Digitalocean.Com/Community/Tutorials/Retrofit-Android-Example-Tutorial.

Beun, R. J., Luiten, C., Verbeek, C., & Poelman, M. P. (2021). A Rationale for a Gamified E-Coach Application to Decrease the Consumption of Sugar Sweetened Beverages. *Frontiers in Digital Health*, *2*. https://doi.org/10.3389/fdgth.2020.564529

Brooke, J. (1995). *SUS: A quick and dirty usability scale*. https://www.researchgate.net/publication/228593520

CNN Indonesia. (2022). *61 Persen Warga Konsumsi Lebih dari 1 Minuman Manis per Hari*. Https://Www.Cnnindonesia.Com/Nasional/20220928191029-20-853924/Kemkes-61-Persen-Warga-Konsumsi-Lebih-Dari-1-Minuman-Manis-per-Hari.

Deva Purwanto, A. G., & Zukhri, Z. (2023). SISTEM INFORMASI PERUMAHAN PURIMAS CITRA GEMILANG 2 BERBASIS MOBILE. *EDUSAINTEK: Jurnal Pendidikan, Sains Dan Teknologi*, *10*(2), 646–658. https://doi.org/10.47668/edusaintek.v10i2.820

Ding, S. (2019). *Smart Tracking for Healthier Lifestyle*. https://scholarworks.rit.edu/theses

Gunawan, A., Handayani, E. T. E., & Purba, O. S. M. (2023). PENGEMBANGAN SISTEM INFORMASI APLIKASI PUSKESMAS UNTUK MENGETAHUI KECUKUPAN GIZI IBU HAMIL DAN MENYUSUI. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *8*(2), 718–728. https://doi.org/10.29100/jipi.v8i2.3650

Hamas, M., & Imaduddin, Z. (2019). PENGEMBANGAN SISTEM JUAL BELI BAHAN POKOK PETANI BERBASIS APLIKASI MOBILE. *Jurnal Informatika Terpadu*, *5*(2), 49–55. https://journal.nurulfikri.ac.id/index.php/JIT

Hasan, M., Alfarizi, J., Candra Brata, K., & Supianto, A. A. (2021). *Pembangunan Sistem Informasi Monitoring dan Evaluasi Pelayanan Kesehatan Orang Dalam Gangguan Jiwa (ODGJ) berbasis Web dengan Metode Waterfall* (Vol. 5, Issue 8). http://j-ptiik.ub.ac.id

Hikmah, N., Suradika, A., Andi, R., & Gunadi, A. (2021). *METODE AGILE UNTUK MENINGKATKAN KREATIVITAS GURU MELALUI BERBAGI PENGETAHUAN (KNOWLEDGE SHARING)*.

IDF. (2021). *Diabetes Atlas (10th edition)*. https://diabetesatlas.org/

Kementerian Kesehatan Republik Indonesia. (2013). *PERATURAN MENTERI KESEHATAN REPUBLIK INDONESIA*.

Kementrian Kesehatan. (2020). *Laporan Riskesdas 2018 Nasional*. https://repository.badankebijakan.kemkes.go.id/id/eprint/3514

Kothari, U., Choudhary, R., Shrivastava, S., Shrivastava, V., Pandey, A., & Sharma, S. (2024). The analysis and effect of Kotlin programming language in modern software development. *AIP Conference Proceedings*, *3214*(1), 020021. https://doi.org/10.1063/5.0239669

Masri, E., Nasution, N. S., & Ahriyasna, R. (2022). Literasi Gizi dan Konsumsi Gula, Garam, Lemak pada Remaja di Kota Padang. *Jurnal Kesehatan*. https://doi.org/10.25047/j-kes

Mauludyani, A. V. R., Nasution, Z., Aries, M., Rimbawan, R., & Egayanti, Y. (2021). Knowledge on Nutrition Labels for Processed Food: Effect on Purchase Decision among Indonesian Consumers. *Jurnal Gizi Dan Pangan*, *16*(1), 47–56. https://doi.org/10.25182/jgp.2021.16.1.47-56

Mulyanto, Y., Karisma, Y., & Maharani, U. (2020). RANCANG BANGUN SISTEM MONITORING PERKEMBANGAN ANAK DI TKIT TAAMASA MEGGUNAKAN METODE SPIRAL. *JINTEKS (Jurnal Informatika Teknologi Dan Sains)*, 190–195. https://doi.org/https://doi.org/10.51401/jinteks.v2i3.754

Pavlidou, E., Papadopoulou, S. K., Seroglou, K., & Giaginis, C. (2023). Revised Harris–Benedict Equation: New Human Resting Metabolic Rate Equation. *Metabolites*, *13*(2). https://doi.org/10.3390/metabo13020189

Plecher, D. A., Eichhorn, C., Steinmetz, C., & Klinker, G. (2020). TrackSugAR. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *12198 LNCS*, 442–459. https://doi.org/10.1007/978-3-030-49904-4_33

Rahmaliani, I., Abdussalaam, F., Gunawan, E., Soelistijaningrum, M., Piksi Ganesha, P., Jend Gatot Subroto No, J., Bandung, K., Barat, J., Al Ihsan, R., Kiastramanggala, J., & Bandung, K. (2023). Tata Kelola Rekam Medis Berbasis Elektronik Dalam Pelaporan

Mortalitas Pasien Rawat Inap Menggunakan Metode Agile Software Development. *JURNAL INOVTEK POLBENG - SERI INFORMATIKA, VOL. 8, NO. 2, 2023*, *8*(2), 2023.

Sari, D. I. P., Restyandito, R., & Nugraha, K. A. (2022). Perancangan Aplikasi Monitoring Kalori Sebagai Upaya Pencegahan Penyakit Diabetes Mellitus Berbasis Android. *Jurnal Teknik Informatika Dan Sistem Informasi*, *8*(1). https://doi.org/10.28932/jutisi.v8i1.4301

Setiawan, P. R. (2020). Sistem Pemesanan Menu Pada Restoran Berbasis Android. *IT Journal Research and Development*, *5*(2), 193–203. https://doi.org/10.25299/itjrd.2021.vol5(2).5866

Sibuea, S., Ikhsan Saputro, M., Annan, A., & Bowo Widodo, Y. (2022). APLIKASI MOBILE COLLECTION BERBASIS ANDROID PADA PT. SUZUKI FINANCE INDONESIA. *JURNAL INFORMATIKA DAN TEKNOLOGI KOMPUTER*, *2*(1).

Suwan, R., Temdee, P., & Prasad, R. (2022). Context-Aware based personalized recommendation on mobile for monitoring excessive sugar consumption of thai adolescents. *Journal of Mobile Multimedia*, *18*(6), 1879–1912. https://doi.org/10.13052/jmm1550-4646.18618

Takemae, A., Takazawa, T., Kamiyama, J., Kanamoto, M., Tobe, M., Hinohara, H., Kunimoto, F., & Saito, S. (2020). A novel prediction equation of resting energy expenditure for Japanese septic patients. *Journal of Critical Care*, *56*, 236–242. https://doi.org/10.1016/j.jcrc.2020.01.021

World Health Organization. (2023). *Diabetes*.

Zuhri, A., Hardiyanti, M., Fitriyani, N. L., & Alfian, G. (2024). Rancang Bangun Aplikasi Monitor Kadar Gula Darah Berbasis Mobile. *Journal of Internet and Software Engineering*, *5*(2), 59–67. https://doi.org/10.22146/jise.v5i2.9026