

**APLIKASI MONITORING KESEHATAN DENGAN
MEMANFAATKAN SMARTWATCH BERBASIS ANDROID**

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana (S1)

ZULFIAN FACHRU REZA

10118170



**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS KOMPUTER INDONESIA
2022**

ABSTRAK

Kesehatan merupakan salah satu hal yang sangat berharga di dalam hidup kita. Dengan tubuh yang sehat, kita sanggup melakukan bermacam-macam pekerjaan dan aktivitas tanpa adanya kendala dan dengan mempunyai raga yang bugar, tentu akan menghasilkan jiwa yang damai juga dan perasaan yang seimbang. Karena itulah menjaga kesehatan tubuh adalah salah satu hal penting yang harus selalu kita upayakan. Terlebih lagi disaat tingginya kasus harian Covid-19 belakangan ini. Salah satu upaya dalam menjaga kesehatan adalah dengan memantau tanda-tanda kesehatan vital, seperti denyut jantung dan tingkat saturasi oksigen dalam darah. Hal yang menjadi poin di dalam penelitian ini adalah pentingnya memantau tanda-tanda vital kesehatan seperti denyut jantung dan kadar oksigen dalam darah terlebih lagi disaat pandemi seperti sekarang dan masih minimnya aplikasi yang dapat memantau kesehatan dengan menggunakan smartwatch yang dapat mengkategorikan data kesehatan tersebut berdasarkan informasi user dan juga memberikan informasi terkait penanganan apabila kondisi dari kesehatan tersebut berada diluar normal. Oleh karena itu, pada tugas akhir kali ini penulis membuat sebuah aplikasi monitoring kesehatan dengan menggunakan smartwatch berbasis android. Aplikasi ini diharapkan mampu membantu orang-orang dalam upaya menjaga kesehatan dengan memantau tanda-tanda kesehatan vital seperti denyut jantung dan saturasi oksigen dan juga mampu memberikan informasi terkait penanganan apabila terdapat kondisi kesehatan yang berada diluar normal.

Kata Kunci: *Aplikasi Monitoring Kesehatan, Denyut Jantung, Saturasi Oksigen*

ABSTRACT

Health is one of the most valuable things in our life. With a healthy body, we are able to do various jobs and activities without any obstacles and have a fit body, of course it will produce a peaceful soul as well and a balanced feeling. That's why maintaining a healthy body is one of the important things that we must always strive for. Especially the high daily cases of Covid-19 lately. One of the efforts in maintaining health is to unite vital health signs, such as heart and oxygen saturation levels in the blood. The point in this research is the importance of integrating vital health signs such as heart rate and blood oxygen levels, especially during a pandemic like now and the lack of applications that integrate health by using a smartwatch that can categorize the health data based on user information and also provide information related to handling if the condition of the health is outside normal. Therefore, in this final project the author makes a health monitoring application using an Android-based smartwatch. This application is expected to be able to help people in an effort to maintain health by unifying vital health signs such as heart and oxygen saturation and also being able to provide information related to handling if there are health conditions that are outside normal.

Keywords: *Health Monitoring App, Heart Rate, Blood Oxygen*

KATA PENGANTAR

Puji dan syukur kami panjatkan ke hadirat Allah SWT, karea atas berkat rahmat dan hidayah-Nya penulis dapat menyelesaikan penulisan tugas akhir skripsi yang berjudul “*Aplikasi Monitoring Kesehatan Dengan Memanfaatkan Smartwatch Berbasis Android*” ini dengan tepat waktu.

Adapun tujuan dari penulisan skripsi ini adalah untuk memenuhi salah satu persyaratan untuk menyelesaikan Program Strata 1 di Universitas Komputer Indonesia. Selain itu, skripsi ini juga bertujuan untuk menambah wawasan tentang pembuatan Aplikasi Monitoring Kesehatan Dengan Memanfaatkan Smartwatch Berbasis Android bagi para pembaca dan juga bagi peneliti.

Dalam penyusunan laporan tugas akhir ini tentulah telah banyak mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu, pada kesempatan ini penulis mengucapkan rasa hormat dan terima kasih kepada

1. Allah SWT yang telah memberikan segalanya, dimulai dari ketabahan, kekuatan, dan kemudahan dalam penyusunan skripsi ini.
2. Kedua orang tua penulis yang selalu memberikan semangat serta do'a kepada penulis selama penyusunan tugas akhir ini.
3. Bapak Ir. Taryana Suryana, M.Kom. selaku dosen pembimbing yang telah meluangkan waktu dan berkenan untuk memberikan bimbingan, arahan serta masukan yang berharga bagi penulis.
4. Kepada keluarga penulis yang tiada hentinya menyemangati penulis dalam penggerjaan tugas akhir ini.
5. Seluruh dosen dan staff pengajar Program Studi Teknik Informatika Universitas Komputer Indonesia.
6. Seluruh teman-teman kelas IF-4 angkatan 2018 atas semangat dan dukungannya.
7. Teman-teman satu bimbingan yang sudah memberikan arahan serta masukan dalam mengerjakan tugas akhir ini.

8. Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu dalam kelancaran penelitian ini, semoga Allah SWT memberikan balasan yang lebih baik.

Penulis menyadari bahwa dalam laporan ini masih banyak kekurangan baik dari segi materi ataupun dalam penyajiannya karena keterbatasan kemampuan dan pengetahuan penulis. Oleh karena itu, kritik dan saran sangat penulis harapkan. Semoga laporan ini dapat bermanfaat bagi pembaca pada umumnya dan penulis pada khususnya. Terima kasih.

Bandung, Agusus 2022

Penulis

DAFTAR ISI

ABSTRAK	i
<i>ABSTRACT</i>	ii
KATA PENGANTAR	iii
DAFTAR ISI.....	v
DAFTAR TABEL.....	ix
DAFTAR GAMBAR	xi
DAFTAR SIMBOL.....	xiv
Use Case Diagram.....	xiv
Activity Diagram.....	xv
Class Diagram	xvi
Sequence Diagram	xvii
DAFTAR LAMPIRAN	xviii
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang Masalah.....	1
1.2. Identifikasi Masalah	2
1.3. Batasan Masalah.....	3
1.4. Maksud dan Tujuan.....	3
1.4.1. Maksud.....	3
1.4.2. Tujuan	3
1.5. Metodologi Penelitian	4
1.5.1. Metode Pengumpulan Data	4
1.5.2. Metode Pembangunan Perangkat Lunak.....	4
1.6. Sistematika Penulisan	5
BAB 2 TINJAUAN PUSTAKA	7

2.1.	Android	7
2.2.	Software Development Kit (SDK)	7
2.3.	Application Programming Interface (API).....	7
2.3.1.	Google Fit API	8
2.4.	Flutter	8
2.4.1.	Kenapa Harus Flutter?.....	9
2.5.	Business Logic Component (BLoC)	10
2.6.	Firebase	11
2.6.1.	Cloud Firestore	11
2.6.2.	Firebase Authentication	12
2.6.3.	Firebase Cloud Messaging	13
2.7.	Denyut Jantung	13
2.7.1.	Takikardia	14
2.7.2.	Bradikardia.....	14
2.7.3.	Pencegahan Denyut Jantung Rendah (Bradikardia).....	15
2.7.4.	Penangan Pertama Kala Alami Denyut Jantung Cepat	15
2.8.	Saturasi Oksigen	16
2.8.1.	Hipoksemia	17
2.8.2.	Happy Hypoxia	17
2.8.3.	Hal Yang Harus Dilakukan Jika Saturasi Oksigen Menurun.....	18
BAB 3	ANALISIS DAN PERANCANGAN SISTEM	19
3.1.	Analisis.....	19
3.1.1.	Analisis Masalah	19
3.1.2.	Analisis Sistem Sedang Berjalan	19
3.1.3.	Analisis Arsitektur Sistem	20

3.1.4.	Analisis Teknologi Yang Digunakan	21
3.1.4.1.	Google Fit API.....	21
3.1.5.	Analisis SKPL.....	22
3.1.5.1.	Analisis Spesifikasi Kebutuhan Fungsional	22
3.1.5.2.	Analisis Spesifikasi Kebutuhan Non-Fungsional	23
3.1.6.	Analisis Kebutuhan Non Fungsional	24
3.1.6.1.	Analisis Kebutuhan Perangkat Keras	24
3.1.6.2.	Analisis Kebutuhan Perangkat Lunak.....	24
3.1.7.	Analisis Kebutuhan Fungsional	25
3.1.7.1.	Use Case Diagram	25
3.1.7.2.	Definisi Aktor	26
3.1.7.3.	Definisi Use Case.....	27
3.1.7.4.	Use Case Scenario	29
3.1.7.5.	Activity Diagram	35
3.1.7.6.	Class Diagram.....	41
3.1.7.7.	Sequence Diagram	42
3.2.	Perancangan	46
3.2.1.	Perancangan Basis Data	46
3.2.2.	Perancangan Struktur Menu.....	47
3.2.3.	Perancangan Antar Muka.....	47
BAB 4	IMPLEMENTASI DAN PENGUJIAN	67
4.1.	Implementasi	67
4.1.1.	Implementasi Perangkat Lunak.....	67
4.1.2.	Implementasi Perangkat Keras.....	68
4.1.3.	Implementasi Basis Data.....	69

4.1.4.	Implementasi Antarmuka	72
4.1.5.	Implementasi Teknologi.....	74
4.1.5.1.	Implementasi Google Fit API.....	74
4.2.	Pengujian.....	76
4.2.1.	Rencana Pengujian Alpha	76
4.2.2.	Hasil Pengujian	77
4.2.3.	Hasil Pengujian User.....	82
	BAB 5 KESIMPULAN DAN SARAN	85
5.1.	Kesimpulan	85
5.2.	Saran.....	85
	DAFTAR PUSTAKA	86

DAFTAR TABEL

Tabel 2. 1 Denyut jantung Berdasarkan Usia.....	13
Tabel 2. 2 Nilai Saturasi Oksigen	16
Tabel 3. 1 Spesifikasi Kebutuhan Fungsional.....	22
Tabel 3. 2 Spesifikasi Kebutuhan Non-Fungsional.....	23
Tabel 3. 3 Definisi Aktor	26
Tabel 3. 4 Definisi Use Case.....	27
Tabel 3. 5 Use Case Scenario UC-01.....	29
Tabel 3. 6 Use Case Scenario UC-02.....	30
Tabel 3. 7 Use Case Scenario UC-03.....	31
Tabel 3. 8 Use Case Scenario UC-04.....	32
Tabel 3. 9 Use Case Scenario UC-05.....	34
Tabel 3. 10 Use Case Scenario UC-06.....	34
Tabel 3. 12 Collection User	46
Tabel 3. 13 Collection Health	47
Tabel 4. 1 Implementasi Perangkat Lunak Laptop	67
Tabel 4. 2 Implementasi Perangkat Lunak Smartphone	68
Tabel 4. 3 Implementasi Perangkat Keras Laptop	68
Tabel 4. 4 Implementasi Perangkat Keras Smartphone	68
Tabel 4. 5 Implementasi User Model.....	70
Tabel 4. 6 Implementasi Health Model.....	71
Tabel 4. 7 Implementasi Antarmuka.....	72
Tabel 4. 8 Rencana Pengujian Alpha	76
Tabel 4. 9 Hasil Pengujian Kasus Login	77
Tabel 4. 10 Hasil Pengujian Kasus Register	78
Tabel 4. 11 Hasil Pengujian Deteksi Denyut Jantung.....	79
Tabel 4. 12 Hasil Pengujian Deteksi Saturasi Oksigen.....	80
Tabel 4. 13 Hasil Pengujian Edit Profil.	81
Tabel 4. 14 Hasil Wawancara dengan Rizky Adha.....	82
Tabel 4. 15 Hasil Wawancara dengan Dian Ayu	83

Tabel 4. 16 Hasil Wawancara dengan Anisa Indriani.....	83
Tabel A. 1 Data Diri Narasumber Pertama	A-1
Tabel A. 2 Data Diri Narasumber Kedua.....	A-4
Tabel A. 3 Data Diri Narasumber Ketiga.....	A-7

DAFTAR GAMBAR

Gambar 1. 1 Tahapan Metode Waterfall.....	5
Gambar 3. 1 Deskripsi Umum Sistem	20
Gambar 3. 2 Menambahkan Permission pada Manifest.....	22
Gambar 3. 3 Use Case Diagram	26
Gambar 3. 4 Activity Diagram UC-01	36
Gambar 3. 5 Activity Diagram UC-02.....	37
Gambar 3. 6 Activity Diagram UC-03	38
Gambar 3. 7 Activity Diagram UC-04.....	39
Gambar 3. 8 Activity Diagram UC-04.....	40
Gambar 3. 9 Activity Diagram UC-06.....	41
Gambar 3. 10 Class Diagram	42
Gambar 3. 11 Sequence Diagram UC-01.....	43
Gambar 3. 12 Sequence Diagram UC-02.....	43
Gambar 3. 13 Sequence Diagram UC-03.....	44
Gambar 3. 14 Sequence Diagram UC-04.....	44
Gambar 3. 15 Sequence Diagram UC-05.....	45
Gambar 3. 16 Sequence Diagram UC-06.....	46
Gambar 3. 17 Struktur Menu	47
Gambar 3. 18 Rancangan Antarmuka Splash Page.....	48
Gambar 3. 19 Rancangan Antarmuka Halaman Login	49
Gambar 3. 20 Rancangan Antarmuka Halaman Register	50
Gambar 3. 21 Rancangan Antarmuka Halaman Home	51
Gambar 3. 22 Rancangan Antarmuka Halaman Home 2	52
Gambar 3. 23 Rancangan Antarmuka Halaman Semua Artikel	53
Gambar 3. 24 Rancangan Antarmuka Halaman Artikel Webview	54
Gambar 3. 25 Rancangan Antarmuka Halaman Kesehatan	55
Gambar 3. 26 Rancangan Antarmuka Halaman Denyut Jantung	56
Gambar 3. 27 Rancangan Antarmuka Halaman Data Denyut Jantung	57
Gambar 3. 28 Rancangan Antarmuka Halaman Tentang Denyut Jantung	58

Gambar 3. 29 Rancangan Antarmuka Halaman Saturasi Oksigen	59
Gambar 3. 30 Rancangan Antarmuka Halaman Data Saturasi Oksigen.....	60
Gambar 3. 31 Rancangan Antarmuka Halaman Tentang Saturasi Oksigen	61
Gambar 3. 32 Rancangan Antarmuka Halaman Penanganan	62
Gambar 3. 33 Rancangan Antarmuka Halaman User	63
Gambar 3. 34 Rancangan Antarmuka Halaman User Edit Nama	64
Gambar 3. 35 Rancangan Antarmuka Halaman User Edit Jenis Kelamin.....	65
Gambar 3. 36 Rancangan Antarmuka Halaman User Edit Tanggal Lahir.....	66
Gambar 4. 1 Menambahkan Package ke Dalam Aplikasi.....	74
Gambar 4. 2 Menambahkan permission baru ke manifest.....	74
Gambar A. 1 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan	A-2
Gambar A. 2 Narasumber sedang mengisi form bukti pengujian.....	A-3
Gambar A. 3 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan	A-5
Gambar A. 4 Narasumber sedang mengisi form bukti pengujian.....	A-6
Gambar A. 5 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan	A-8
Gambar A. 6 Narasumber sedang mengisi form bukti pengujian.....	A-9
Gambar B. 1 Screenshot Halaman Splash Page.....	B-1
Gambar B. 2 Screenshot Halaman Login.....	B-2
Gambar B. 3 Screenshot Halaman Register.....	B-3
Gambar B. 4 Screenshot Halaman Home	B-4
Gambar B. 5 Screenshot Halaman Home 2	B-5
Gambar B. 6 Screenshot Halaman Artikel.....	B-6
Gambar B. 7 Screenshot Halaman Artikel Webview.....	B-7
Gambar B. 8 Screenshot Halaman Kesehatan.....	B-8
Gambar B. 9 Screenshot Halaman Denyut Jantung	B-9
Gambar B. 10 Screenshot Halaman Data Denyut Jantung	B-10
Gambar B. 11 Screenshot Halaman Tentang Denyut Jantung	B-11
Gambar B. 12 Screenshot Halaman Saturasi Oksigen	B-12

Gambar B. 13 Screenshot Halaman Data Saturasi Oksigen	B-13
Gambar B. 14 Screenshot Halaman Tentang Saturasi Oksigen.....	B-14
Gambar B. 15 Screenshot Halaman Penanganan Denyut Jantung Rendah	B-15
Gambar B. 16 Screenshot Halaman Penanganan Denyut Jantung Tinggi	B-16
Gambar B. 17 Screenshot Halaman Penanganan Saturasi Oksigen Rendah	B-17
Gambar B. 18 Screenshot Halaman User.....	B-18
Gambar B. 19 Screenshot Halaman User Edit Nama	B-19
Gambar B. 20 Screenshot Halaman User Edit Jenis Kelamin	B-20
Gambar B. 21 Screenshot Halaman User Edit Tanggal Lahir	B-21
Gambar C. 1 Hasil Review di Playstore.....	C-1
Gambar C. 2 Hasil Review di Playstore.....	C-2

DAFTAR SIMBOL

Use Case Diagram

No	Gambar	Nama	Keterangan
1		Actor	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case
2		Use Case	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
3		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
4		Include	Hubungan dimana objek anak (descendent) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (ancestor).
5		Extend	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
6		System	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

Activity Diagram

No	Gambar	Nama	Keterangan
1		Activity	State dari sistem yang mencerminkan eksekusi dari suatu aksi
2		Initial Node	Menunjukkan dimulainya suatu aktifitas.
3		Decision	Digunakan untuk menggambarkan suatu pengambilan keputusan / tindakan pada kondisi tertentu
4		Final Node	Menunjukkan akhir dari aktifitas.

Class Diagram

No	Gambar	Nama	Keterangan
1		Class	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
2	-	Private Visibility	Merupakan simbol yang menunjukkan bahwa method atau atribut tersebut hanya bisa diakses oleh kelasnya saja
3	+	Public Visibility	Merupakan simbol yang menunjukkan bahwa method atau atribut tersebut bisa diakses oleh kelas lainnya yang terhubung dengan kelas tersebut.
4	#	Protected Visibility	Merupakan simbol yang menunjukkan bahwa method atau atribut tersebut bisa diakses oleh kelasnya sendiri dan turunannya saja
5		Composition	Mengindikasikan bahwa suatu kelas menggunakan atau mengandung object yang berada di kelas lain yang terhubung

Sequence Diagram

No	Gambar	Nama	Keterangan
1		Lifeline	Objek entity, antarmuka yang saling berinteraksi.
2		Message	Mengindikasikan komunikasi antar objek.
3		Reply Message	Mengindikasikan sebuah balasan pesan antar objek yang saling berkomunikasi.

DAFTAR LAMPIRAN

Lampiran A Hasil Wawancara	A-1
Lampiran B Screenshot Aplikasi	B-1
Lampiran C Hasil Review di Google Playstore	C-1
Lampiran D Listing Program	D-1

BAB 1

PENDAHULUAN

1.1. Latar Belakang Masalah

Kesehatan merupakan salah satu hal yang sangat berharga di dalam hidup kita. Dengan tubuh yang sehat, kita sanggup melakukan bermacam-macam pekerjaan dan kesibukan tanpa adanya kendala dan dengan mempunyai raga yang bugar, tentu akan menghasilkan jiwa yang damai dan perasaan yang seimbang. Karena itulah menjaga kesehatan tubuh adalah salah satu hal penting yang harus selalu kita upayakan. Apalagi disaat tingginya kasus harian Covid-19 belakangan ini, alangkah baiknya kita harus meningkatkan kewaspadaan kita dengan menerapkan protokol kesehatan yang telah dianjurkan pemerintah. Salah satu upaya dalam menjaga kesehatan adalah dengan memantau tanda-tanda kesehatan vital, seperti detak jantung dan tingkat saturasi oksigen dalam darah [1].

Jantung merupakan organ yang vital, penting dan esensial. Fungsinya memengaruhi hampir seluruh organ tubuh yang lain. Jika kita mempunyai permasalahan pada jantung maka kelangsungan hidup kita juga akan terancam. Oleh karena itu, penting untuk menjaga kesehatan jantung untuk hidup yang lebih berkualitas [2]. Detak jantung atau denyut nadi adalah berapa kali jantung kamu berdetak dalam 1 menit. Aktivitas yang kamu lakukan bisa mengubah seberapa cepat atau lambat detak jantung berfluktuasi, mulai dari detak lambat dan stabil saat beristirahat atau tidur, hingga detak jantung yang cepat selama kamu berolahraga. Detak jantung normal orang dewasa dengan anak-anak berbeda. Jika detak jantung normal orang dewasa berkisar antara 60-100 detak per menit, anak-anak biasanya memiliki detak jantung yang lebih tinggi. Dengan mengetahui detak jantung normal pada anak-anak dan orang dewasa, kamu bisa mengetahui kondisi kesehatan jantung kamu dan anak lebih baik lagi, sehingga bisa segera mencari penanganan bila menemukan adanya kejanggalan [3].

Selain detak jantung, ada saturasi oksigen yang tak kalah penting yang harus tetap harus kita jaga. Saturasi oksigen merupakan nilai yang menunjukkan kadar oksigen di dalam darah. Nilai ini sangat berpengaruh terhadap

berbagai fungsi organ dan jaringan tubuh. Pengukuran nilai saturasi oksigen dapat dilakukan dengan beberapa cara, yakni dengan analisis gas darah (AGD), menggunakan alat oximeter atau yang sekarang lagi hype yaitu menggunakan *smartwatch*. Meskipun *smartwatch* tak bisa dianggap sebagai alat medis, namun hasil yang didapatkan dinilai cukup akurat mendekati standar alat medis. Nilai saturasi oksigen normal pada orang dengan kondisi paru-paru yang sehat atau tidak memiliki kondisi medis tertentu memiliki nilai SpO₂ 95-100%. Sementara itu, pada orang yang memang memiliki penyakit paru-paru seperti Penyakit Paru Obstruktif Kronik (PPOK), memiliki nilai saturasi oksigen yang berbeda, berada pada 88-92% [4].

Berdasarkan uraian permasalahan di atas, dibutuhkan solusi yang dapat membantu orang-orang dapat memantau tanda-tanda kesehatan vital, seperti detak jantung, dan saturasi oksigen. Maka dari itu akan dibangun sebuah aplikasi yang dapat memonitoring kesehatan berbasis android dengan memanfaatkan *smartwatch*. Pada aplikasi tersebut, sistem akan mencatat data-data seperti kondisi detak jantung, nilai saturasi oksigen, dari pengguna setiap kali pengguna tersebut mengeceknya menggunakan *smartwatch*. Dari data-data yang sudah didapatkan, nantinya pengguna akan mendapatkan informasi terkait kondisi kesehatannya. Seperti adanya kelainan pada detak jantung atau tidak, dan menampilkan nilai saturasi oksigen apakah berada pada kondisi normal atau tidak. Nantinya, aplikasi akan memberikan informasi terkait penanganan sesuai dengan kondisi kesehatan yang didapatkan.

1.2. Identifikasi Masalah

Berdasarkan latar belakang di atas, maka didapatkan masalah yang dapat diidentifikasi, yaitu adalah sebagai berikut :

1. Pentingnya memantau tanda-tanda vital kesehatan seperti detak jantung dan saturasi oksigen dalam darah terlebih di saat pandemi seperti ini sebagai upaya untuk meningkatkan kewaspadaan terhadap Covid-19.
2. Masih minimnya aplikasi yang dapat memantau kesehatan dengan menggunakan *smartwatch* yang langsung memberikan informasi terkait

penanganan pertama ataupun saran atas masalah dari kondisi kesehatan tersebut.

1.3. Batasan Masalah

Adapun batasan masalah dalam penelitian ini, yaitu sebagai berikut:

1. Sistem yang dibangun hanya akan mendukung sistem operasi Android dengan versi *sdk* minimal 5.0 *lollipop* atau API Level 21
2. Smartwatch yang digunakan di dalam penelitian ini, yaitu Samsung Galaxy Watch 4.
3. Sistem yang dibangun hanya akan berjalan jika terdapat aplikasi Google Fit pada *smartphone* yang digunakan.
4. Database yang digunakan yaitu Firebase Cloud Firestore.
5. Untuk autentikasinya yaitu menggunakan Firebase Authentication.
6. Untuk layanan push notification-nya menggunakan Firebase Cloud Messaging (FCM)
7. Aplikasi harus terkoneksi dengan internet.

1.4. Maksud dan Tujuan

Adapun maksud dan tujuan dari penelitian ini adalah sebagai berikut :

1.4.1. Maksud

Maksud dari penelitian ini adalah untuk membangun aplikasi berbasis android yang dapat memonitoring kesehatan dengan menggunakan *smartwatch*.

1.4.2. Tujuan

Adapun tujuan yang akan dicapai dalam penelitian ini diantaranya sebagai berikut :

1. Membantu orang-orang dalam upaya menjaga kesehatan dengan memantau tanda-tanda kesehatan vital seperti detak jantung dan saturasi oksigen.

- Memberikan informasi terkait penanganan pertama jika terdapat kondisi kesehatan diluar normal.

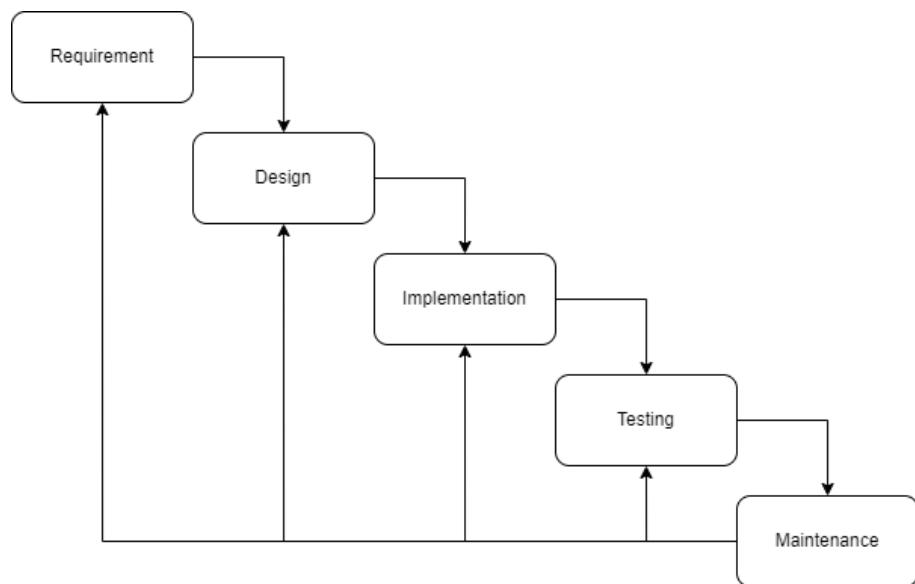
1.5. Metodologi Penelitian

1.5.1. Metode Pengumpulan Data

Tahap pertama yang dilakukan pada metode penelitian ini adalah pengumpulan data atau informasi. Untuk pengumpulan informasi ini sendiri, akan dilakukan dengan beberapa cara seperti mencari informasi melalui media buku, jurnal, atau bahkan dari situs web kesehatan seperti Halodoc, Alodokter, WHO, dan situs web kesehatan resmi lainnya.

1.5.2. Metode Pembangunan Perangkat Lunak

Metode pembangunan perangkat lunak yang digunakan pada penelitian ini adalah metode *waterfall*. Model *waterfall* ini pertama kali muncul sekitar tahun 1970 sehingga sering dianggap kuno, namun merupakan model yang paling banyak dipakai di dalam software engineering. Saat ini model *waterfall* merupakan model pengembangan perangkat lunak yang sering digunakan. Model pengembangan ini melakukan pendekatan secara sistematis dan berurutan. Disebut *waterfall* karena tahap demi tahap yang dilalui harus menunggu selesainya tahap sebelumnya dan berjalan berurutan. Model pengembangan ini bersifat linear dari tahap awal pengembangan sistem yaitu tahap perencanaan sampai tahap akhir pengembangan sistem yaitu tahap pemeliharaan. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan dan tidak bisa kembali atau mengulang ke tahap sebelumnya [15]. Tahapan dari metode *waterfall* ini dapat dilihat pada gambar 1.1



Gambar 1. 1 Tahapan Metode Waterfall

1.6. Sistematika Penulisan

Untuk memudahkan penulisan dalam penyusunan skripsi ke arah yang dimaksud, maka digunakan sistematika penulisan laporan yang nantinya digunakan untuk mempermudah pembahasan.

BAB I PENDAHULUAN

Bab ini berisi tentang latar belakang, identifikasi masalah, Batasan masalah, maksud dan tujuan, dan sistematika penulisan. Pada bab ini menjelaskan permasalahan yang sedang terjadi dan menjelaskan solusi untuk masalah tersebut.

BAB II TINJAUAN PUSTAKA

Bab ini membahas teori-teori pendukung yang digunakan dalam proses perencanaan, proses perancangan dan proses pembuatan pada penelitian, seperti landasan teori terkait android, flutter, BLoC, firebase, denyut jantung, saturasi oksigen dan teori lainnya yang berhubungan dengan penelitian.

BAB III ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan berisi tentang analisis, dan perancangan sistem pada penelitian. Bab ini bertujuan untuk memudahkan pembaca atau peneliti

selanjutnya dalam memahami fungsionalitas apa saja yang dimiliki oleh sistem atau aplikasi.

BAB IV IMPLEMENTASI DAN PENGUJIAN SISTEM

Bab ini membahas implementasi dalam bahasa pemrograman yaitu implementasi kebutuhan perangkat keras dan perangkat lunak, implementasi basis data, implementasi antarmuka dan tahap - tahap dalam melakukan pengujian perangkat lunak.

BAB V KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan yang sudah diperoleh dari hasil penulisan tugas akhir dan saran-saran untuk memperbaiki sistem di penelitian berikutnya.

BAB 2

TINJAUAN PUSTAKA

2.1. Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencangkup sistem operasi, middleware, dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel / smartphone. Android merupakan generasi baru platform mobile yang memberikan pengembangan untuk melakukan pengembangan sesuai dengan yang diharapkannya. Sistem operasi yang mendasari Android dilisensikan dibawah GNU, General Public Lisensi versi 2 (GPLv2), yang sering dikenal dengan istilah “copyleft” lisensi dimana setiap perbaikan pihak ketiga harus terus jatuh dibawah terms. Android didistribusikan di bawah lisensi Apache Software (ASL/Apache2), yang memungkinkan untuk distribusi kedua dan seterusnya.

2.2. Software Development Kit (SDK)

SDK adalah seperangkat alat yang dapat digunakan untuk membuat dan mengembangkan aplikasi. Secara umum, SDK mengacu pada modul *software suite* lengkap yang mencakup semua yang Anda butuhkan untuk modul tertentu dalam suatu aplikasi. *Tools* atau alat SDK akan mencakup berbagai hal, termasuk *library*, dokumentasi, contoh kode, proses, dan panduan yang dapat digunakan dan diintegrasikan oleh pengembang ke dalam aplikasi mereka sendiri. SDK dirancang untuk digunakan untuk platform atau bahasa pemrograman tertentu.

2.3. Application Programming Interface (API)

API adalah singkatan dari *Application Programming Interface*, yaitu sebuah software yang memungkinkan para developer untuk mengintegrasikan dan

mengizinkan dua aplikasi yang berbeda secara bersamaan untuk saling terhubung satu sama lain.

Tujuan penggunaan API adalah untuk saling berbagi data antar aplikasi yang berbeda. Selain itu API juga bertujuan mempercepat proses pengembangan aplikasi dengan cara menyediakan sebuah fungsi terpisah sehingga para developer tidak perlu lagi membuat fitur yang serupa.

Istilah API sebetulnya tidak ada hubungannya dengan hal-hal yang berkaitan dengan web, karena istilah tersebut sudah ada sebelum web. Hal ini semacam dikooptasi yang berarti “pemanggilan web service”.

2.3.1. Google Fit API

Google Fit merupakan salah satu platform pelacakan kesehatan yang dikembangkan oleh Google untuk berbagai sistem operasi, baik android, maupun wear os. Google Fit API merupakan API yang digunakan untuk mendapatkan data kesehatan yang ada pada Google Fit. Data kesehatan yang digunakan dalam aplikasi meliputi data denyut jantung dan juga saturasi oksigen.

2.4. Flutter

Flutter adalah sebuah bahasa pemrograman buatan Google pada tahun 2011 yang populer pada akhir-akhir ini. Flutter adalah sebuah platform yang banyak dipakai oleh mobile developer untuk pembuatan aplikasi multiplatform menggunakan satu basis coding. Yang berarti flutter adalah sebuah teknologi open source berasal dari Google yang digunakan untuk pembuatan aplikasi pada sistem operasi Android dan iOS.

Flutter adalah sebuah Software Development Kit (SDK) yang membantu para mobile developer, sehingga sudah pasti dilengkapi dengan berbagai macam fitur berguna buat mengembangkan aplikasi pada lintas platform.

Contoh pada rendering engine, widget flutter adalah yang siap pakai, pengujian dan integrasi pada API, serta command-line tools. Meskipun ada desain serupa berupa teknologi seperti react native tetapi para mobile developer tetap memakai flutter karena kemudahan yang ditawarkan. Inilah salah satu faktor

kenapa mobile developer suka flutter, yaitu kemudahan membuat aplikasi dengan menggunakan satu basis kode saja.

Hal tersebut sesuai dengan klaim dari pihak Google sebagai pembuat aplikasi. Jika kamu sebagai mobile developer dan ingin mengembangkan aplikasi berbasis flutter, maka kamu harus menguasai bahasa pemrograman dart terlebih dahulu.

Hubungan dart dan flutter adalah dart merupakan bahasa pemrograman pada flutter yang berfokus untuk mengembangkan front end sehingga dapat dimanfaatkan dalam pembuatan aplikasi mobile dan website.

2.4.1. Kenapa Harus Flutter?

a. Cross Platform

Flutter mendukung cross platform alias dapat dijalankan di beberapa platform yang berbeda. Dengan menggunakan Flutter, kita dapat membuat aplikasi Android dan iOS sekaligus. Selain mobile, kita juga dapat membuat aplikasi web dan desktop. Tentunya hal ini akan menghemat waktu. Kita tidak perlu mempelajari bahasa native yang digunakan di masing-masing platform.

b. Fast Development (Hot Reload)

Flutter memiliki sebuah fitur bernama hot reload. Dengan fitur ini, proses pengembangan aplikasi dapat berjalan lebih cepat dan mudah.. Setelah melakukan perubahan pada kode program, cukup tekan hot reload. Aplikasi akan diperbarui dalam kurun waktu kurang dari 1 detik. Sangat cepat bukan?

Hot reload bekerja dengan cara menginjeksi kode program yang mengalami perubahan ke dalam Dart Virtual Machine. Setelah virtual machine memperbarui tiap kelas dengan kode program versi terbaru, maka framework Flutter secara otomatis membangun kembali susunan komponen widget sehingga kita dapat dengan cepat melihat perubahan yang terjadi.

c. Beautiful UI

Flutter dirancang untuk mempermudah developer dalam membangun tampilan user interface. Keseluruhan UI pada Flutter dibangun menggunakan widget. Sebagai contoh, jika kita menambahkan sebuah text field, text field tersebut adalah widget. Button dan Image juga merupakan widget. Bahkan untuk mengatur posisi komponen menjadi rata tengah, kita menggunakan center widget. Kita dapat melakukan kustomisasi pada tiap widget. Widget akan menggambarkan seperti apa tampilan yang akan dibuat berdasarkan konfigurasi dan state yang ada. Terdapat 2 set widget, Material Design (Android) dan Cupertino (iOS). Teman-teman dapat melihat daftar lengkap widget yang tersedia di tautan ini. Selain itu, di tiap pekan, tim Flutter juga membahas Widget of the Week yang dapat teman-teman saksikan lewat channel Youtube Flutter.

2.5. Business Logic Component (BLoC)

BLoC (Business Logic Component) adalah suatu pattern architecture yang dikenalkan oleh Google pada tahun lalu. Dengan dimana ini merupakan cara untuk memisahkan bisnis components dengan logic yang memudahkan untuk membagi ke beberapa Dart aplikasi. Jadi dengan menggunakan architecture ini diharapkan dari sisi pengembangan akan lebih flexible bedasarkan kebutuhan masing – masing proses.

Banyak dari state management menjadi solusi dari flutter, tapi kita harus memilih salah satu dari state. Kemungkinan untuk mendapatkan state management yang sempurna sesuai dengan kita sangatlah susah bahkan terkesan tidak ada. Tetapi yang dibutuhkan adalah suatu State Management yang terbaik sesuai dengan kebutuhan team dan project.

Bloc dikembangkan dengan berdasarkan

1. Simple : mudah di mengerti dan di gunakan oleh banyak developer dari berbagai skill.
2. Powerful: hal ini membuat suatu aplikasi yang komplek dapat di kemas dalam component – component yang kecil

3. Testable : memudahkan tes di setiap component aplikasi, dikarenakan pemisahan setiap aktivitas menjadi suatu layer

2.6. Firebase

Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para developer aplikasi dalam mengembangkan aplikasinya. Firebase alias BaaS (*Backend as a Service*) merupakan solusi yang ditawarkan oleh Google untuk mempercepat pekerjaan developer.

Dengan menggunakan Firebase, apps developer bisa fokus dalam mengembangkan aplikasi tanpa memberikan *effort* yang besar untuk urusan *backend*.

Singkat cerita mengenai sejarah dari Firebase didirikan pertama kali pada tahun 2011 oleh Andrew Lee dan James Tamplin. Produk Firebase yang pertama kali adalah Realtime Database. Realtime Database digunakan developer untuk menyimpan data dan *synchronize* ke banyak *user*. Kemudian ia berkembang sebagai layanan pengembang aplikasi. Pada bulan Oktober 2014, perusahaan tersebut diakuisisi oleh Google.

Mengenai segi layanan, dulu Firebase memberikan *service trial* (percobaan), namun saat ini kamu bisa memanfaatkan dan menggunakan layanan Firebase secara *free* (gratis). Tentu saja dengan adanya batasan-batasan tertentu.

2.6.1. Cloud Firestore

Cloud Firestore adalah database yang fleksibel dan skalabel untuk pengembangan seluler, web, dan server dari Firebase dan Google Cloud Platform. Seperti Firebase Realtime Database, Cloud Firestore membuat data Anda tetap sinkron di semua aplikasi klien melalui pemroses realtime, serta menawarkan dukungan offline untuk seluler dan web. Dengan begitu, Anda dapat mem-build aplikasi yang responsif dan mampu berfungsi tanpa bergantung pada latensi jaringan atau konektivitas Internet. Cloud Firestore juga menawarkan integrasi

yang lancar dengan produk Firebase dan Google Cloud lainnya, termasuk Cloud Functions.

Cloud Firestore adalah database NoSQL yang dihosting di cloud dan dapat diakses langsung oleh aplikasi Apple, Android, dan web Anda melalui SDK native. Cloud Firestore juga tersedia di SDK native Node.js, Java, Python, Unity, C++, dan Go, selain REST API dan RPC API.

Sesuai dengan model data NoSQL Cloud Firestore, Anda menyimpan data dalam dokumen yang berisi kolom yang dipetakan terhadap nilai. Dokumen ini disimpan dalam koleksi, yang merupakan container untuk dokumen Anda, yang dapat digunakan untuk mengatur data dan mem-build kueri. Dokumen mendukung berbagai jenis data, mulai dari string dan angka sederhana, hingga objek bertingkat yang kompleks. Anda juga dapat membuat subkoleksi dalam dokumen dan mem-build struktur data hierarkis yang dapat diskalakan sesuai dengan pertumbuhan database. Model data Cloud Firestore mendukung struktur data yang paling sesuai untuk aplikasi Anda.

2.6.2. Firebase Authentication

Sebagian besar aplikasi perlu mengetahui identitas pengguna. Dengan mengetahui identitas pengguna, aplikasi dapat menyimpan data pengguna secara aman di cloud dan memberikan pengalaman personal yang sama di setiap perangkat pengguna.

Firebase Authentication menyediakan layanan backend, SDK yang mudah digunakan, dan library UI siap pakai untuk mengautentikasi pengguna ke aplikasi Anda. Firebase Authentication juga mendukung autentikasi menggunakan sandi, nomor telepon, serta penyedia identitas gabungan yang populer seperti Google, Facebook, Twitter, dan lain-lain.

Firebase Authentication terintegrasi erat dengan layanan Firebase lainnya dan sistem ini memanfaatkan berbagai standar industri, seperti OAuth 2.0 dan OpenID Connect, sehingga dapat dengan mudah diintegrasikan dengan backend kustom Anda.

2.6.3. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) adalah solusi pertukaran pesan lintas platform yang dapat diandalkan untuk mengirim pesan tanpa biaya. Dengan FCM, kita dapat memberi tahu aplikasi klien bahwa email baru atau data lainnya tersedia untuk disinkronkan. Kita dapat mengirim pesan notifikasi untuk mendorong interaksi kembali dan retensi pengguna. Untuk kasus penggunaan seperti instant messaging, pesan dapat mentransfer payload hingga 4.000 byte ke aplikasi klien.

2.7. Denyut Jantung

Jantung adalah organ vital yang bertugas untuk memompa darah ke seluruh tubuh. Hal ini membuat peredaran darah yang kaya oksigen dapat mencapai seluruh sel-sel dalam tubuh Anda. Untuk mengecek kesehatan jantung, dokter biasanya akan mengamati seberapa normal tekanan darah dan detak jantung.

Detak jantung atau yang sering disebut juga sebagai denyut nadi adalah berapa kali jantung Anda berdetak per menit. Detak jantung atau denyut nadi yang normal berkisar antara 60 hingga 100 detak per menit (BPM). Denyut jantung normal yang dikategorikan berdasarkan usia dapat dilihat pada tabel 2.1 berikut.

Tabel 2. 1 Denyut jantung Berdasarkan Usia

Usia	Denyut Jantung (bpm)
Bayi	70 – 190
Anak - anak	80 – 110
Remaja	60 - 100
Dewasa	60 – 100

Secara medis, gangguan irama jantung disebut dengan istilah aritmia. Kondisi ini ditandai dengan denyut jantung yang terlalu cepat, lambat, tidak teratur, atau bahkan terhenti sama sekali.

Aritmia dapat disebabkan oleh berbagai hal, seperti riwayat penyakit jantung, tekanan darah tinggi, penyakit katup jantung, gangguan tiroid, gangguan elektrolit, atau sedang dalam masa pemulihan setelah menjalani operasi jantung. Gaya hidup tidak sehat, seperti konsumsi minuman beralkohol secara berlebihan dan kebiasaan merokok, serta efek samping obat-obatan, juga dapat menyebabkan aritmia.

Penyakit aritmia secara garis besar dibagi menjadi dua, yaitu takikardia dan bradikardia.

2.7.1. Takikardia

Takikardia adalah kondisi ketika detak jantung berdetak lebih cepat saat istirahat. Belum diketahui secara pasti penyebab kondisi ini, tetapi ada beberapa faktor yang diduga dapat memicu takikardia.

Faktor tersebut meliputi faktor keturunan, riwayat penyakit tertentu seperti penyakit jantung dan anemia, efek samping obat-obatan, atau kebiasaan seperti merokok dan mengonsumsi minuman beralkohol.

Takikardia dapat menimbulkan keluhan berupa nyeri dada, pusing, kelelahan, dan sesak napas. Namun, ada kalanya takikardia tidak menimbulkan gejala atau keluhan apa pun.

2.7.2. Bradikardia

Detak jantung yang terlalu lambat disebut bradikardia. Normalnya, jantung berdetak 60–100 kali per menit ketika istirahat. Namun, pada kondisi bradikardia, detak jantung kurang dari 60 kali dalam satu menit.

Kondisi ini dapat dipengaruhi oleh faktor pertambahan usia, kebiasaan merokok, efek samping obat-obatan, atau riwayat penyakit seperti tekanan darah tinggi atau kelainan tiroid.

Pada sebagian orang, mungkin detak jantung yang terlalu lambat tidak menimbulkan masalah. Namun, kondisi ini bisa menjadi tanda adanya masalah pada sistem listrik jantung.

Bradikardia dapat menimbulkan keluhan berupa sesak napas, sulit konsentrasi, pingsan, pusing, dan mudah lelah walau hanya melakukan sedikit aktivitas.

2.7.3. Pencegahan Denyut Jantung Rendah (Bradikardia)

Melambatnya denyut jantung umumnya merupakan hal yang normal. Kondisi tersebut dapat terjadi pada orang yang sedang tidur, remaja, atau atlet. Namun, jika disertai dengan gejala pusing atau sesak nafas, denyut jantung yang melambat bisa menjadi tanda adanya gangguan pada aktivitas listrik jantung

Bradikardia dapat dicegah dengan menghindari faktor-faktor yang dapat meningkatkan risiko terjadinya kondisi ini. Caranya adalah dengan mengubah gaya hidup agar lebih sehat, yaitu dengan melakukan langkah sederhana berikut ini:

1. Menghindari kebiasaan merokok.
2. Menghindari penggunaan NAPZA
3. Membatasi konsumsi alkohol
4. Menghindari stress
5. Menjaga berat badan ideal
6. Berolahraga secara rutin
7. Mengonsumsi makanan bergizi seimbang dan rendah garam

2.7.4. Penangan Pertama Kala Alami Denyut Jantung Cepat

Terkadang denyut jantung sangat cepat (di atas 100 kali per menit) seseorang alami walau tanpa sebab pasti dan ini bisa saja merupakan gangguan irama jantung. Spesialis jantung sub spesialis aritmia dari RS Jantung dan Pembuluh Darah, Harapan Kita, Prof. Dr. dr. Yoga Yunradi, Sp.JP(K) menyarankan sembari mencari pertolongan medis ada beberapa upaya sederhana yang bisa dilakukan untuk mengatasinya sementara. Hal yang pertama dilakukan yaitu bisa menaruh kain berisi air es di leher atau membasuh muka dengan air es untuk menhentikan sirkuit. Selain itu, tutup kedua lubang hidung sambal menahan napas beberapa detik. Kendati begitu, menurut Yoga cara ini hanya menawarkan

tingkat keberhasilan sekitar 40 persen. Namun, saat debaran jantung bersamaan dengan penurunan kesadaran, sebaiknya bawa penderita ke rumah sakit untuk mendapatkan pemeriksaan dan perawatan

2.8. Saturasi Oksigen

Saturasi oksigen merupakan nilai yang menunjukkan kadar oksigen di dalam darah. Nilai ini sangat berpengaruh terhadap berbagai fungsi organ dan jaringan tubuh. Pengukuran nilai saturasi oksigen dapat dilakukan dengan 2 cara, yakni dengan analisis gas darah (AGD) atau menggunakan alat oximeter.

Analisis gas darah adalah metode pengukuran saturasi oksigen yang dilakukan dengan cara mengambil sampel darah dari pembuluh darah arteri. Hasil analisis gas darah sangat akurat, karena pengukurnya dilakukan di rumah sakit dan dikerjakan oleh tenaga medis profesional.

Sementara itu, oximeter adalah alat pengukur saturasi oksigen yang berbentuk klip. Pengukurnya dilakukan dengan cara menjepitkan oximeter pada jari tangan. Saturasi oksigen kemudian akan diukur berdasarkan jumlah cahaya yang dipantulkan oleh sinar inframerah, yang dikirim ke pembuluh darah kapiler.

Berbeda dengan analisis gas darah, pengukuran saturasi oksigen dengan oximeter bisa dilakukan sendiri dengan mudah di rumah. Oximeter bahkan kini direkomendasikan oleh Badan Kesehatan Dunia (WHO) untuk dimiliki di setiap rumah guna mengukur nilai saturasi oksigen secara berkala.

Hasil pengukuran saturasi oksigen yang dilakukan dengan analisis gas darah ditunjukkan dengan istilah PaO₂ (tekanan parsial oksigen). Sementara itu, hasil pengukuran saturasi oksigen dengan menggunakan oximeter ditunjukkan dengan istilah SpO₂.

Berikut merupakan tabel untuk nilai saturasi oksigen.

Tabel 2. 2 Nilai Saturasi Oksigen

Nilai SpO ₂	Kategori
95 – 100%	Normal

$< 95\%$	Rendah
----------	--------

Orang yang memiliki saturasi oksigen rendah atau hipoksemia bisa merasakan berbagai gejala, seperti nyeri dada, sesak napas, batuk, sakit kepala, detak jantung cepat, kebingungan, dan kulit membiru.

Kendati demikian, orang yang mengalami hipoksemia juga bisa tidak merasakan gejala apa pun. Kondisi ini yang disebut dengan *happy hypoxia* ini bisa terjadi pasien COVID-19.

Hipoksemia, baik yang menimbulkan gejala maupun tidak, bisa menganggu kerja organ dan jaringan tubuh. Bila dibiarkan, hal ini dapat menyebabkan kerusakan pada organ vital, seperti jantung, otak, dan ginjal, dan berisiko menyebabkan komplikasi yang berbahaya.

2.8.1. Hipoksemia

Hipoksemia adalah kondisi di mana kadar oksigen di dalam darah di bawah batas normal. Padahal, oksigen sangat diperlukan untuk menjaga organ dan jaringan tubuh, termasuk jantung, otak, ginjal, dan lainnya, agar tetap berfungsi dengan baik. Hipoksemia bisa terdeteksi melalui pemeriksaan fisik serta tes darah.

2.8.2. Happy Hypoxia

Istilah *happy hypoxia* digunakan untuk menunjukkan kondisi berkurangnya kadar oksigen di dalam tubuh tanpa menimbulkan gejala. Meski sulit dikenali, kondisi ini perlu diwaspadai karena dapat berakibat fatal, terutama bagi penderita COVID-19.

Hingga saat ini, penyebab *happy hypoxia* belum diketahui secara pasti. Namun, ada teori yang menyebutkan bahwa *happy hypoxia* terjadi akibat peradangan pada jaringan paru-paru yang disebabkan oleh infeksi virus Corona.

2.8.3. Hal Yang Harus Dilakukan Jika Saturasi Oksigen Menurun

Normalnya, kadar oksigen dalam darah berkisar 95 sampai 100 persen, yang menandakan organ seperti jantung, paru-paru, dan peredaran darah berfungsi baik. Namun, tidak perlu panik jika saturasi oksigen berada di bawah normal. Pasalnya ketika kadar oksigen mengalami penurunan, ada beberapa untuk meningkatkan saturasi oksigen yang dapat dilakukan sebagai pertolongan pertama.

Langkah pertama yang bisa dilakukan jika mengalami saturasi oksigen rendah adalah dengan melakukan pengecekan ulang saturasi oksigen dengan menggunakan oximeter atau menggunakan smartwatch dalam kondisi jari yang tidak basah.

Kedua adalah dengan melakukan prone atau tengkurep dengan meletekkan bantal atau guling di pergelangan kaki agar punggung terasa rileks. Berdasarkan hasil penelitian posisi ini dapat meningkatkan oksigenasi tubuh. Lakukan prone selama 30 menit.

Ketiga adalah posisikan tubuh anda setengah duduk, bisa dilakukan di tempat tidur. Hal ini bisa anda lakukan selama 30 menit.

Keempat adalah Latihan pengembangan dada atau chest expansion dengan cara menarik nafas melalui hidung dan buang lewat mulut. Lakukan secara perlahan 10 sampai 15 kali. Lakukan 4 hal ini secara bergantian dan anda bisa kembali mengecek saturasi oksigen kembali.

Apabila sudah melakukan hal-hal yang disarankan namun belum bisa meningkatkan nilai saturasi oksigen dalam tubuh, segera ke Fasilitas Pelayanan Kesehatan (Fasyankes) terdekat untuk mendapatkan penanganan medis lebih lanjut.

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis

Tahap analisis bertujuan untuk menganalisis permasalahan-permasalahan yang ada serta menentukan kebutuhan apa saja yang dibutuhkan oleh sistem yang akan dibangun. Analisis yang akan dibahas analisis masalah, analisis sistem sedang berjalan, analisis arsitektur sistem, analisis teknologi yang digunakan, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional. Analisis ini bertujuan untuk membuat sistem yang akan dibangun dapat menyelesaikan semua permasalahan yang ada di BAB 1.

3.1.1. Analisis Masalah

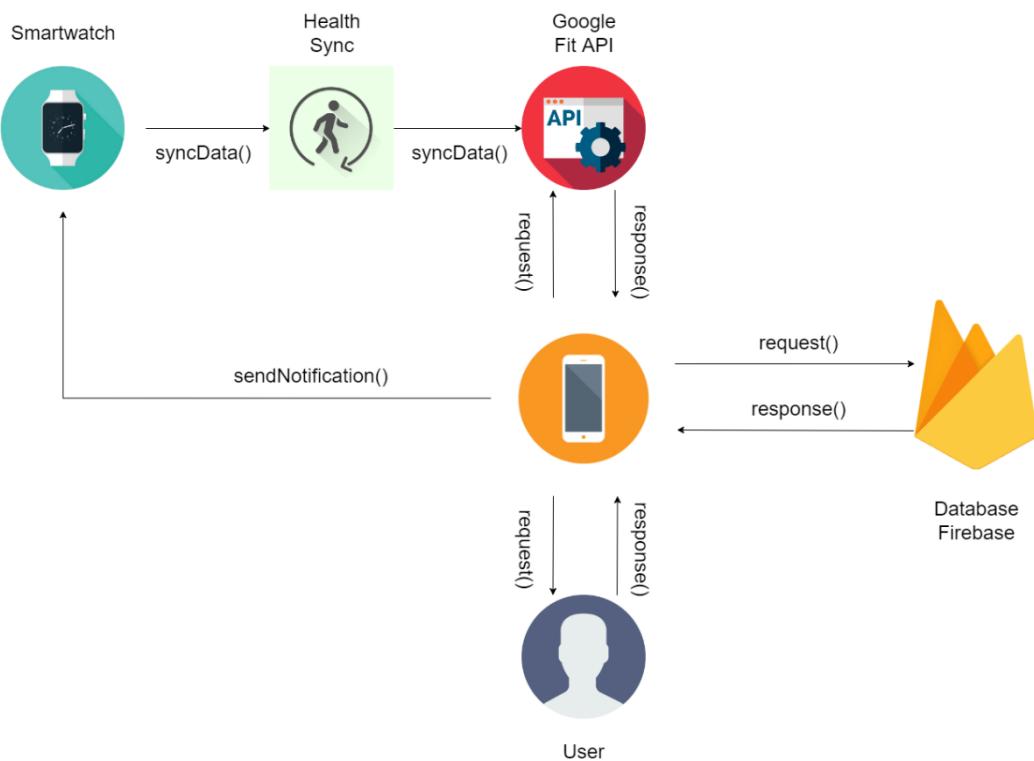
Pada tahap ini akan dijelaskan masalah yang ditemui dan akan menjadi sebuah dasar dalam pembuatan aplikasi. Seperti yang kita tahu, kesehatan merupakan salah satu hal yang sangat berharga di dalam hidup kita. Dengan tubuh yang sehat, kita sanggup melakukan pekerjaan dan kesibukan tanpa adanya kendala. Salah satu upaya dalam menjaga kesehatan adalah dengan memantau tanda-tanda kesehatan vital seperti detak jantung dan saturasi oksigen. Akan tetapi, saat ini belum terdapat aplikasi yang dapat memberikan informasi terkait kondisi kesehatannya dan juga memberikan informasi tips dan juga penanganan pertama jika teridentifikasi kondisi kesehatannya dibawah rata-rata.

3.1.2. Analisis Sistem Sedang Berjalan

Sebelum membangun suatu aplikasi, tahap awal yang wajib dilakukan adalah mempelajari dan menganalisis aplikasi yang ada atau sedang berjalan. Saat ini kebanyakan aplikasi hanya menambahkan data kesehatan lalu melihat riwayat penambahan data tanpa adanya informasi apakah data tersebut termasuk normal atau diluar normal dan juga tidak diberikannya informasi terkait penanganan terhadap kondisi kesehatannya jika teridentifikasi berada diluar normal.

3.1.3. Analisis Arsitektur Sistem

Arsitektur dibuat untuk mendefinisikan komponen-komponen yang ada dalam sistem secara lebih spesifik. Pada penelitian ini, untuk arsitektur sistemnya dapat dilihat pada gambar 3.1



Gambar 3. 1 Deskripsi Umum Sistem

Berikut penjelasan alur dari gambar deskripsi umum sistem pada gambar 3.1

1. User mengakses smartphone
2. Pastikan smartphone sudah terhubung dengan smartwatch
3. User melakukan pengecekan kesehatan, contohnya denyut jantung pada smartwatch.
4. Singkronisasi data antara smartwatch dan juga Google Fit menggunakan aplikasi pihak ketiga, yaitu health sync.
5. Smartphone melakukan request kepada Google Fit API untuk mendapatkan data kesehatan.
6. Google Fit API akan mengirimkan data yang diminta ke smartphone (kemudian data tersebut disimpan ke database firebase)

7. Smartphone akan mengoleh data kesehatan tersebut, dan jika terjadi kondisi data kesehatan diluar normal maka smartphone akan mengirimkan notification ke smartwatch.
8. Smartwatch menerima notifikasi berdasarkan kondisi yang diperoleh.

3.1.4. Analisis Teknologi Yang Digunakan

Analisis teknologi dimaksudkan untuk memberi gambaran teknologi apa saja yang digunakan di dalam penelitian ini.

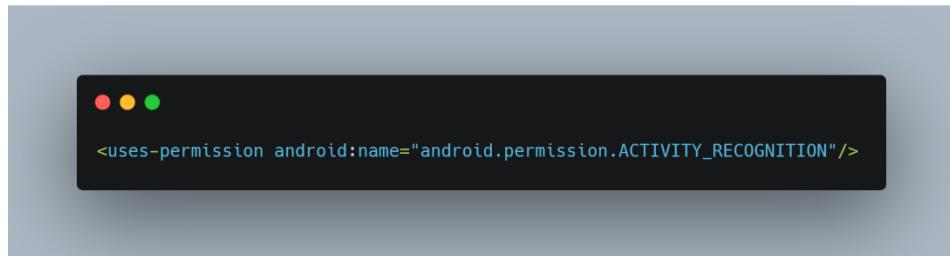
3.1.4.1. Google Fit API

Google Fit API digunakan untuk mendapatkan data kesehatan yang ada pada Google Fit. Data kesehatan yang digunakan dalam aplikasi meliputi data denyut jantung dan juga saturasi oksigen.

Untuk mengimplementasikan Google Fit API ke project Flutter, Langkah-langkah yang diperlukan adalah sebagai berikut.

1. Karena disini kita akan menggunakan flutter sebagai teknologi utama dalam pembuatan aplikasinya, maka kita akan menggunakan sebuah package yang dapat membantu kita dalam penggunaan Google Fit API ini. Package tersebut yaitu health, dengan package ini kita sangat dibantu dalam pengaplikasian Google Fit API ke dalam project kita. Untuk dokumentasi lebih lengkapnya bisa dilihat melalui tautan <https://pub.dev/packages/health>.
2. Hal yang selanjutnya harus dilakukan adalah mendapatkan API key, yang dapat didapatkan melalui panduan berikut <https://developers.google.com/fit/android/get-api-key>. Itu panduan yang ada di dalam tautan tersebut untuk mendapatkan API key.
3. Langkah yang selanjutnya adalah mengaktifkan layanan Google Fit API, melalui tautan <https://console.cloud.google.com/apis/enableflow?apiid=fitness>. Lalu pilih Enable API and Service.

4. Langkah selanjutnya adalah membuat kredensial baru dengan tipe OAuth Client ID.
5. Langkah selanjutnya, yaitu menambahkan permission baru pada file manifest. Untuk permission-nya dapat dilihat pada gambar 3.2.



Gambar 3. 2 Menambahkan Permission pada Manifest

Setelah semua langkah telah dilakukan, maka Google Fit API sudah siap digunakan pada project flutter kita.

3.1.5. Analisis SKPL

Analisis SKPL atau Spesifikasi Kebutuhan Perangkat Lunak merupakan tahapan yang menjelaskan hasil dari analisis kebutuhan-kebutuhan fungsional apa saja yang harus ada di dalam sistem agar dapat memecahkan masalah-masalah yang ada. Selain kebutuhan fungsional, tahapan ini juga membahas kebutuhan non-fungsional.

3.1.5.1. Analisis Spesifikasi Kebutuhan Fungsional

Berdasarkan dari analisis masalah, maka didapatkan spesifikasi kebutuhan fungsional yang dapat dapat dilihat pada tabel 3.1.

Tabel 3. 1 Spesifikasi Kebutuhan Fungsional

No	SKPL-ID	Spesifikasi Kebutuhan Fungsional
1	SKPL-F01	Sistem memiliki fitur login untuk bisa masuk ke dalam aplikasi utama
2	SKPL-F02	Sistem memiliki fitur register untuk membuat akun agar bisa masuk ke dalam aplikasi utama

3	SKPL-F03	Sistem dapat mengambil data denyut jantung dan mengklasifikasikannya sebagai normal, rendah, atau tinggi berdasarkan data umur dan juga jenis kelamin.
4	SKPL-F04	Sistem dapat mengambil data saturasi oksigen dan mengklasifikasikannya sebagai normal atau rendah.
5	SKPL-F05	Sistem dapat mengirimkan notifikasi ke smartwatch apabila nilai denyut jantung atau saturasi oksigen berada diluar normal.
6	SKPL-F06	Sistem dapat mengubah profil apabila terjadi kesalahan saat memasukan data saat proses registrasi
7	SKPL-F07	Sistem dapat mengakhiri sesi login dari aplikasi

3.1.5.2. Analisis Spesifikasi Kebutuhan Non-Fungsional

Untuk kebutuhan non-fungsional, dapat dilihat pada tabel 3.2.

Tabel 3. 2 Spesifikasi Kebutuhan Non-Fungsional

No	SKPL-ID	Spesifikasi Kebutuhan Non-Fungsional
1	SKPL-NF01	Sistem yang dibuat berbasis mobile android
2	SKPL-NF02	Teknologi utama yang digunakan untuk pembuatan sistem yaitu Flutter SDK yang menggunakan Bahasa pemrograman Dart.
3	SKPL-NF03	Sistem menggunakan Firebase sebagai Backend as a Service yang meng-handle beberapa hal seperti penyimpanan menggunakan layanan Cloud Firestore, autentikasi menggunakan Firebase Authentication, dan layanan <i>push notification</i>

		menggunakan Firebase Cloud Messaging.
4	SKPL-NF04	Sistem yang dibangun akan berjalan pada sistem operasi android dengan versi sdk minimal 5.0 lollipop atau API level 21.

3.1.6. Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

3.1.6.1. Analisis Kebutuhan Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk membangun aplikasi adalah sebagai berikut :

- a. Prosesor : Amd A8
- b. RAM : 8 GB RAM
- c. Harddisk : 500GB
- d. Keyboard
- e. Mouse

3.1.6.2. Analisis Kebutuhan Perangkat Lunak

Spesifikasi perangkat lunak pendukung yang dibutuhkan dalam pembangunan aplikasi yaitu :

- 1. Sistem Operasi : Windows 11
- 2. IDE : Android Studio

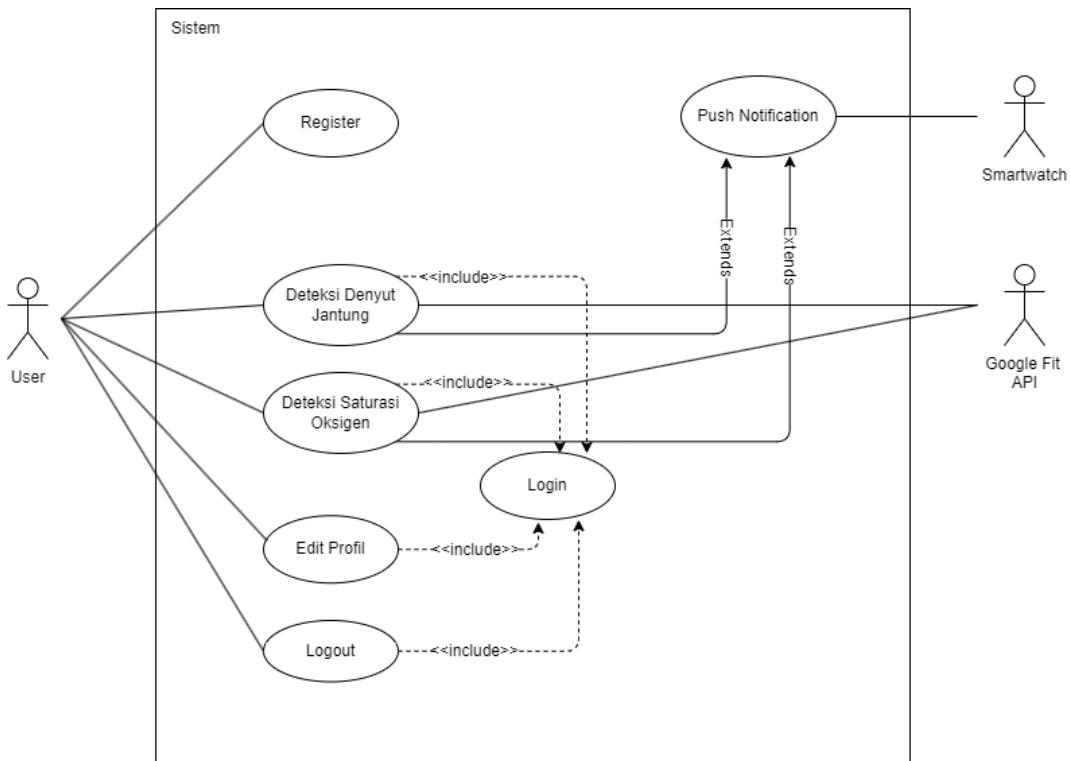
3. Code Editor : Visual Studio Code
4. Google Chrome
5. Genymotion
6. Postman

3.1.7. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional dapat mendefinisikan sebagai gambaran, perencanaan dan pembuatan sketsa atau pengaturan dari beberapa elemen untuk dijadikan satu kesatuan yang utuh dan berfungsi. Tahap ini menyangkut konfigurasi dari komponen perangkat lunak dan perangkat keras dari suatu sistem sehingga instalasi dari sistem bisa berfungsi. Adapun tahapan analisis sistem menggunakan UML meliputi Use Case Diagram, Use Case Scenario, Activiy Diagram, Class Diagram dan Sequence Diagram.

3.1.7.1. Use Case Diagram

Use Case Diagram adalah diagram yang menggambarkan hubungan interaksi antara sistem dan actor. Use case dapat mendeskripsikan tipe interaksi antara si pengguna sistem dengan sistemnya. Berikut use case diagram untuk sistem yang akan dibangun dapat dilihat pada gambar 3.3.



Gambar 3. 3 Use Case Diagram

3.1.7.2. Definisi Aktor

Tabel 3. 3 Definisi Aktor

No	Aktor	Deskripsi
1	User	Aktor dengan role ini memiliki wewenang untuk login dan register ke dalam sistem, melakukan deteksi denyut jantung, deteksi saturasi oksigen dan memprediksi kemungkinan adanya penyakit jantung. User juga dapat mengedit profil dan melakukan logout dari aplikasi.
2	Smartwatch	Aktor dengan role ini memiliki wewenang untuk menerima notifikasi yang dikirimkan oleh aplikasi
3	Google Fit API	Aktor dengan role ini memiliki wewenang sebagai penyedia API / sumber data.

3.1.7.3. Definisi Use Case

Tabel 3. 4 Definisi Use Case

No	Use Case	Deskripsi
1	Register	<p>Kode Use Case: UC-01</p> <p>Deskripsi Singkat: Sistem menyediakan halaman register untuk pembuatan akun agar bisa masuk ke dalam sistem.</p> <p>Aktor: User</p> <p>Trigger: User memasukan data-data yang telah disediakan, diantarnya nama, email, password, jenis kelamin dan tanggal lahir dan menekan tombol daftar.</p> <p>Prakondisi: Halaman register ditampilkan.</p> <p>Pascakondisi: Data user berhasil disimpan dan user langsung diarahkan langsung ke halaman utama.</p>
2	Login	<p>Kode Use Case: UC-02</p> <p>Deskripsi Singkat: Sistem menyediakan halaman login sebelum bisa masuk ke dalam sistem</p> <p>Aktor: User</p> <p>Trigger: User memasukkan data email dan password lalu menekan tombol masuk.</p> <p>Prakondisi: Halaman login ditampilkan.</p> <p>Pascakondisi: User yang datanya telah terdaftar akan berhasil masuk dan langsung diarahkan ke halaman utama.</p>
3	Deteksi Denyut Jantung	<p>Kode Use Case: UC-03</p> <p>Deskripsi Singkat: Sistem mengambil data denyut jantung ke Google Fit API lalu</p>

		<p>mengklasifikasikannya sebagai normal, rendah atau tinggi.</p> <p>Aktor: User</p> <p>Trigger: User memilih icon sync atau sinkronisasi data pada halaman denyut jantung.</p> <p>Prakondisi: Halaman denyut jantung akan ditampilkan.</p> <p>Pascakondisi: Sistem mengklasifikasi data denyut jantung.</p>
4	Deteksi Saturasi Oksigen	<p>Kode Use Case: UC-04</p> <p>Deskripsi Singkat: Sistem mengambil data saturasi oksigen ke Google Fit API lalu mengklasifikasikannya sebagai normal atau rendah.</p> <p>Aktor: User</p> <p>Trigger: User memilih icon sync atau sinkronisasi data pada halaman saturasi oksigen.</p> <p>Prakondisi: Halaman saturasi oksigen akan ditampilkan.</p> <p>Pascakondisi: Sistem mengklasifikasi data saturasi oksigen.</p>
5	Edit Profile	<p>Kode Use Case: UC-05</p> <p>Deskripsi Singkat: Sistem menyediakan halaman untuk user bisa mengedit profilnya.</p> <p>Aktor: User</p> <p>Trigger: User memilih data mana yang akan diubah.</p> <p>Prakondisi: Halaman user akan ditampilkan</p>

		Pascakondisi: Sistem akan mengubah data profil dari user sesuai dengan data yang diubah dan menyimpannya ke database
6	Logout	<p>Kode Use Case: UC-06</p> <p>Deskripsi Singkat: Sistem akan mengakhiri sesi login.</p> <p>Aktor: User</p> <p>Trigger: Menekan tombol logout</p> <p>Prakondisi: Halaman user ditampilkan.</p> <p>Pascakondisi: User berhasil logout dan akan langsung diarahkan ke halaman login.</p>

3.1.7.4. Use Case Scenario

Pada bagian ini akan diisi dengan scenario untuk setiap use case dengan menggambarkan urutan interaksi aktor dengan use case tersebut dari awal sampai akhir. Untuk use case scenario yang pertama, yaitu use scenario untuk use case dengan code UC-01 dapat dilihat pada tabel 3.5.

Tabel 3. 5 Use Case Scenario UC-01

Use Case Name	Register
Goal in Context	Pengguna berhasil mendaftarkan akunnya untuk bisa masuk ke dalam sistem.
Description	Fungsionalitas ini digunakan oleh user untuk melakukan pendaftaran akun agar bisa masuk ke dalam sistem aplikasi
Related Use Case	UC-01
Successful End Condition	User berhasil mendaftarkan akunnya.
Failed End	User tidak berhasil mendaftarkan akunnya.

Condition		
Actors	User	
Trigger	User mengisi data-data yang telah disediakan dan menekan tombol daftar.	
Main Flow	Step	Action
	1	User membuka aplikasi
	2	Sistem akan menampilkan halaman login
	3	User menekan link daftar
	4	Sistem menampilkan halaman register
	5	User menekan tombol daftar
	6	Sistem akan mengecek apakah data yang dikirimkan sudah benar
	7	User berhasil melakukan register dan data disimpan ke dalam database.
	8	Sistem akan mengarahkan user ke halaman utama

Untuk use case scenario selanjutnya, yaitu usecase scenario untuk use case dengan kode UC-02 dapat dilihat pada tabel 3.6.

Tabel 3. 6 Use Case Scenario UC-02

Use Case Name	Login	
Goal in Context	Pengguna berhasil masuk ke dalam sistem menggunakan akun yang sebelumnya sudah terdaftar.	
Description	Fungsionalitas ini digunakan oleh user untuk masuk ke dalam sistem menggunakan akun yang sebelumnya sudah terdaftar.	
Related Use Case	UC-01, UC-03, UC-04, UC-05, UC-08, UC-09	
Successful Condition	User berhasil masuk ke dalam sistem.	

Failed Condition	End	User tidak berhasil masuk ke dalam sistem.
Actors		User
Trigger		User mengisi email dan password lalu menekan tombol masuk.
Main Flow	Step	Action
	1	User membuka aplikasi
	2	Sistem akan menampilkan halaman login
	3	User menekan tombol login
	4	Sistem akan mengecek apakah data yang dimasukkan terdaftar pada database.
	5	User berhasil masuk ke dalam sistem.
	6	Sistem akan mengarahkan user ke halaman utama

Untuk use case scenario selanjutnya, yaitu use scenario untuk use case dengan kode UC-03 dapat dilihat pada tabel 3.7.

Tabel 3. 7 Use Case Scenario UC-03

Use Case Name	Deteksi Denyut Jantung
Goal in Context	Sistem berhasil mengklasifikasikan data denyut jantung, mengirimkan notifikasi, dan memberikan informasi terkait penanganan
Description	Fungsionalitas ini digunakan untuk mengklasifikasikan data denyut jantung lalu mengirimkan notifikasi dan memberikan informasi terkait penanganan apabila terdapat kondisi denyut jantung yang berada diluar normal.
Related Use Case	UC-03
Successful End	Sistem berhasil mengklasifikasikan data denyut jantung.

Condition		
Failed End Condition	User tidak berhasil mengklasifikasikan data denyut jantung.	
Actors	User	
Trigger	User membuka halaman denyut jantung dan menekan tombol sync	
Main Flow	Step	Action
	1	User membuka aplikasi
	2	User memilih menu kesehatan
	3	User membuka halaman denyut jantung
	4	Sistem akan menampilkan halaman denyut jantung
	5	User menekan tombol sync
	6	Sistem akan mengambil data denyut jantung ke Google Fit API
	7	Sistem akan mengklasifikasikan data denyut jantung.
	8	Sistem akan mengirimkan notifikasi dan memberikan informasi penanganan apabila terdapat kondisi denyut jantung yang berada diluar normal.

Untuk use case scenario selanjutnya, yaitu use scenario untuk use case dengan kode UC-04 dapat dilihat pada tabel 3.8.

Tabel 3. 8 Use Case Scenario UC-04

Use Case Name	Deteksi Saturasi Oksigen
Goal in Context	Sistem berhasil mengklasifikasikan data saturasi oksigen, mengirimkan notifikasi, dan memberikan informasi terkait penanganan

Description	Fungsionalitas ini digunakan untuk mengklasifikasikan dat saturasi oksigen lalu mengirimkan notifikasi dan memberikan informasi terkait penanganan apabila terdapat kondisi saturasi oksigen yang berada diluar normal.	
Related Use Case	UC-04	
Successful End Condition	Sistem berhasil mengklasifikasikan data saturasi oksigen.	
Failed End Condition	User tidak berhasil mengklasifikasikan data saturasi oksigen.	
Actors	User, Google Fit API	
Trigger	User membuka halaman saturasi oksigen dan menekan tombol sync	
Main Flow	Step	Action
	1	User membuka aplikasi
	2	User memilih menu kesehatan
	3	User membuka halaman saturasi oksigen
	4	Sistem akan menampilkan halaman saturasi oksigen
	5	User menekan tombol sync
	6	Sistem akan mengambil data saturasi oksigen ke Google Fit API
	7	Sistem akan mengklasifikasikan data saturasi oksigen.
	8	Sistem akan mengirimkan notifikasi dan memberikan informasi penanganan apabila terdapat kondisi saturasi oksigen yang berada diluar normal.

Untuk use case scenario selanjutnya, yaitu use case scenario untuk use case dengan kode UC-05 dapat dilihat pada tabel 3.9.

Tabel 3. 9 Use Case Scenario UC-05

Use Case Name	Edit Profile	
Goal in Context	User dapat mengubah data profilnya.	
Description	Fungsionalitas ini digunakan oleh user mengubah data profilnya apabila terdapat kesalahan pada saat proses pendaftaran	
Related Use Case	UC-05	
Successful End Condition	User berhasil mengubah data profil.	
Failed End Condition	User tidak berhasil mengubah data profil.	
Actors	User	
Trigger	User membuka halaman user dan memilih data mana yang akan diubah.	
Main Flow	Step	Action
	1	User membuka halaman utama
	2	User memilih menu akun
	3	Sistem akan menampilkan halaman user
	4	User memilih data mana yang akan diubah
	5	User mengubah data yang sebelumnya sudah dipilih
	6	Sistem berhasil mengubah data profil dan menyimpannya ke dalam database.

Untuk use case scenario selanjutnya, yaitu use case scenario untuk use case dengan kode UC-06 dapat dilihat pada tabel 3.10.

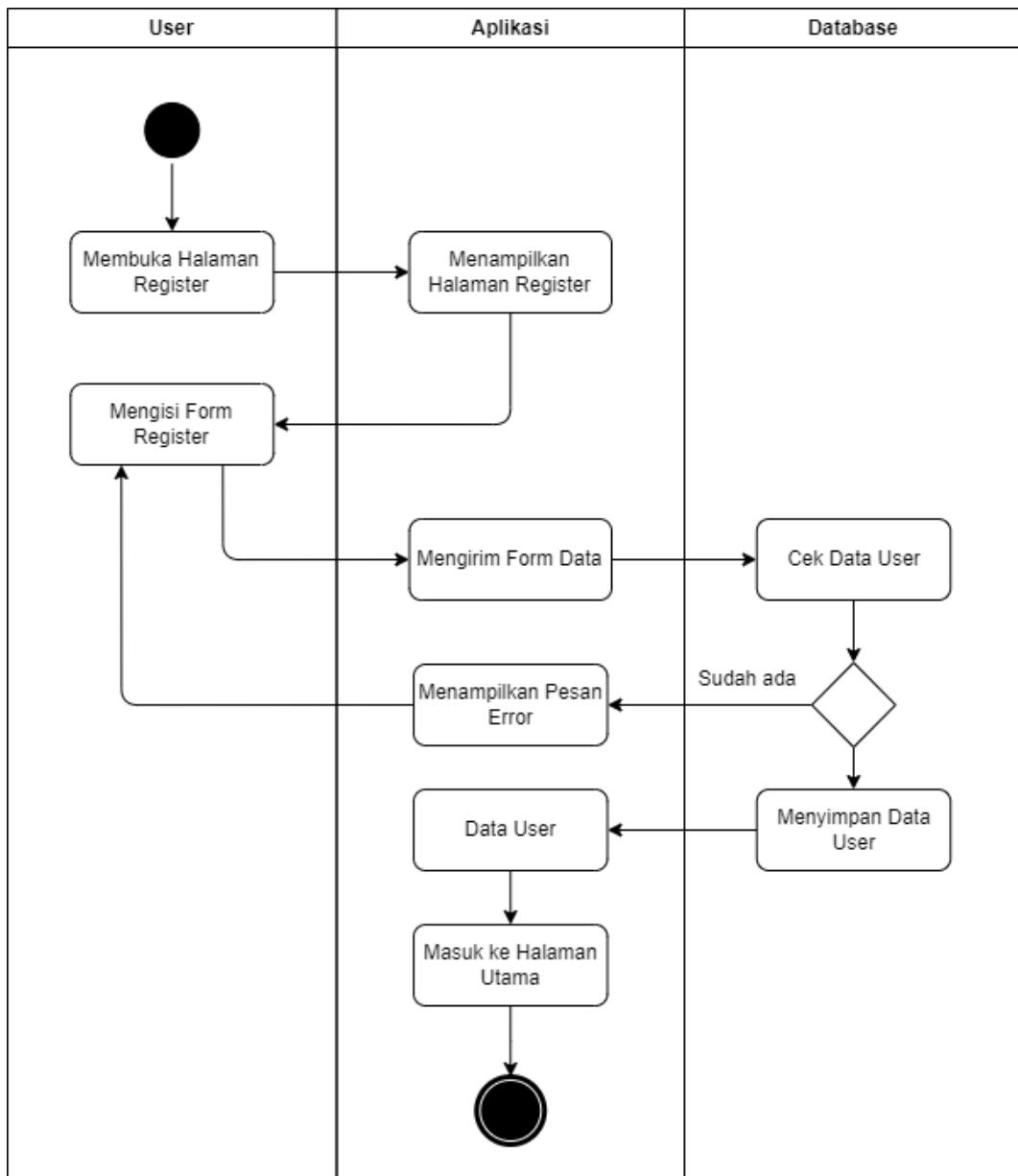
Tabel 3. 10 Use Case Scenario UC-06

Use Case Name	Logout
---------------	--------

Goal in Context	User keluar dari aplikasi.	
Description	Fungsionalitas ini digunakan oleh user untuk keluar dari aplikasi dan mengakhiri sesi login	
Related Use Case	UC-06	
Successful Condition	End	User berhasil keluar dari aplikasi dan mengakhiri sesi login.
Failed Condition	End	User tidak berhasil keluar dari aplikasi dan mengakhiri sesi login.
Actors	User	
Trigger	User membuka halaman user dan menekan tombol logout.	
Main Flow	Step	Action
	1	User membuka halaman utama
	2	User memilih menu akun
	3	Sistem akan menampilkan halaman user
	4	User menekan tombol logout
	5	Sistem akan mengakhiri sesi login dari user
	6	Sistem akan mengarahkan user kembali ke halaman login

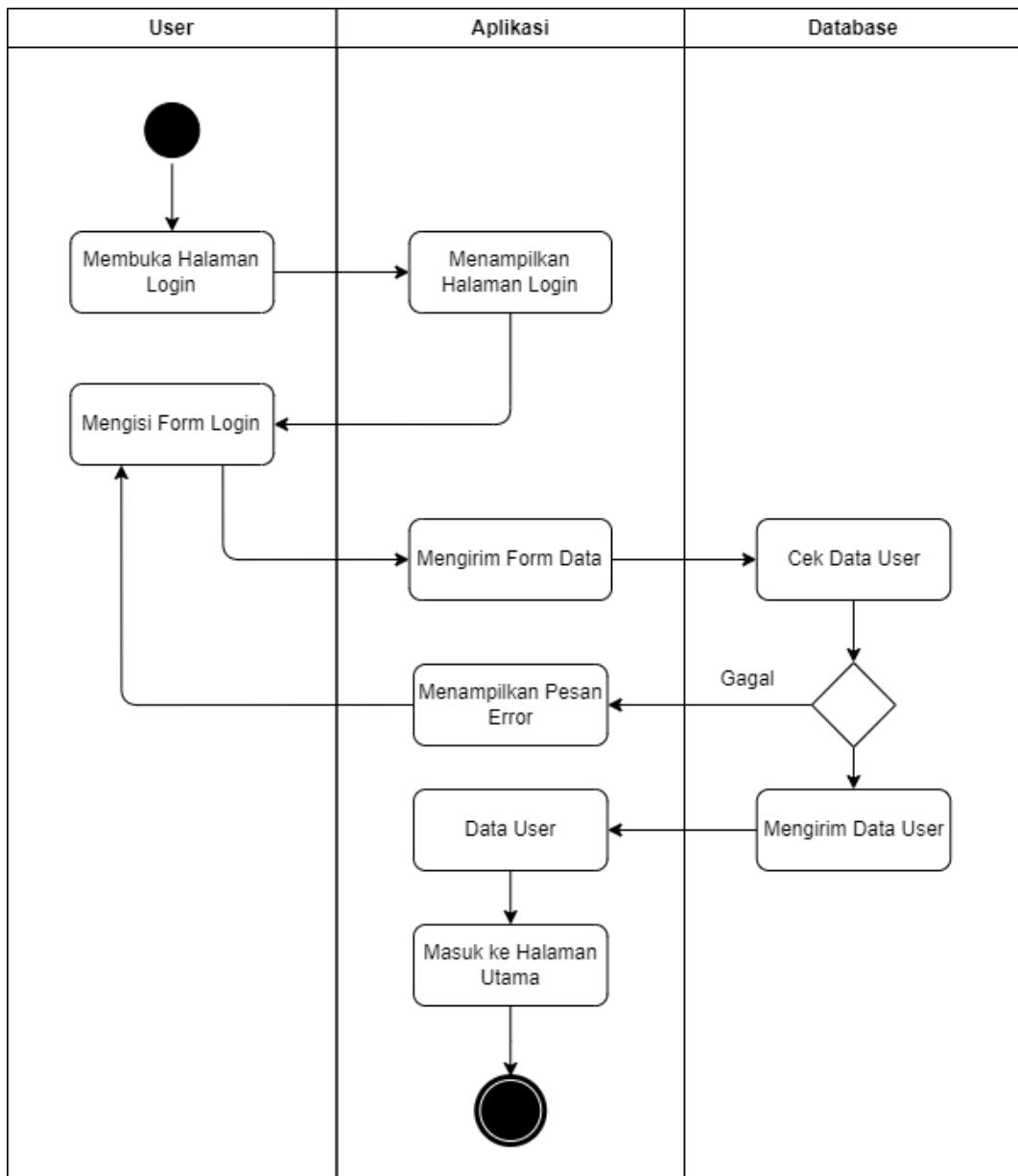
3.1.7.5. Activity Diagram

Activity diagram merupakan penggambaran proses dari aktivitas yang dilakukan. Activity diagram untuk use case dengan kode UC-01 dapat dilihat pada gambar 3.4.



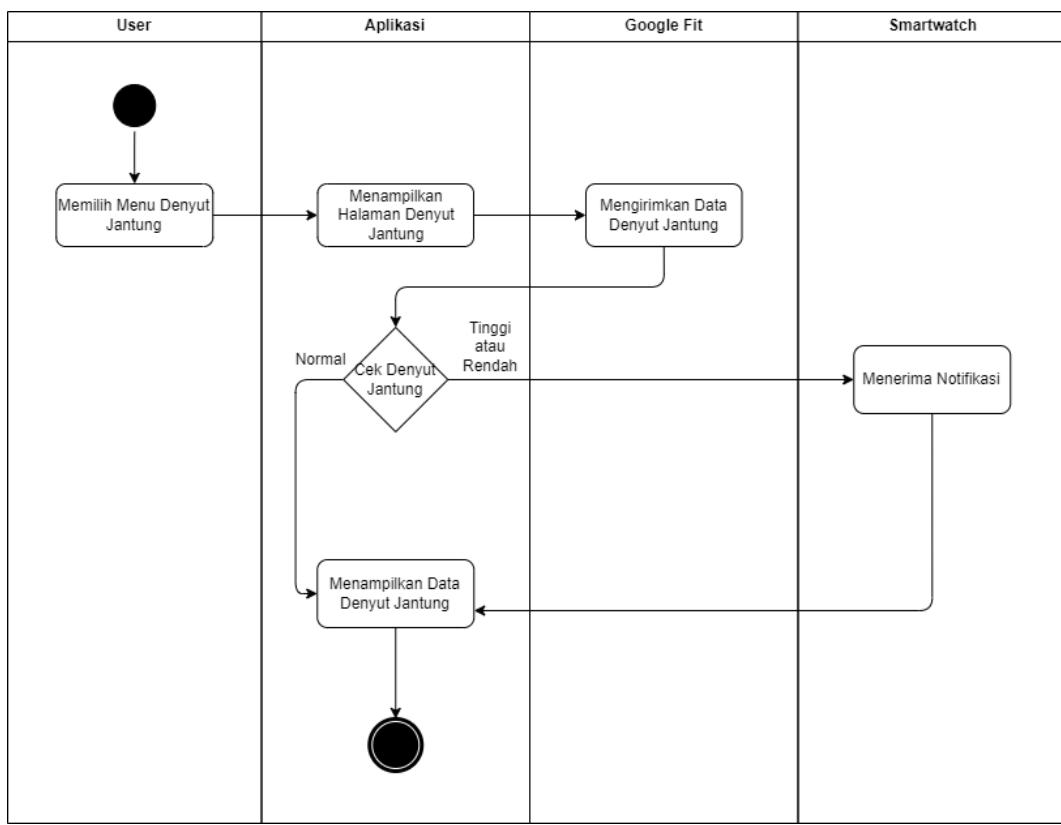
Gambar 3. 4 Activity Diagram UC-01

Untuk activity diagram dengan kode use case UC-02 dapat dilihat pada gambar 3.5.



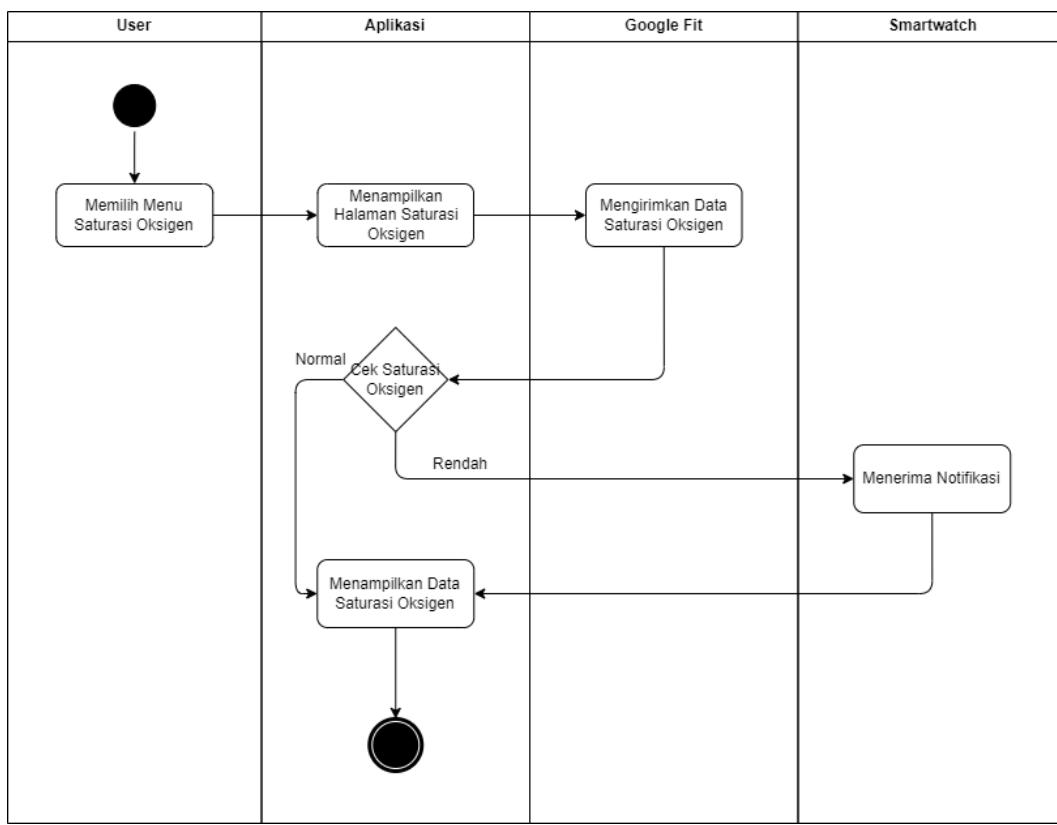
Gambar 3. 5 Activity Diagram UC-02

Untuk activity diagram dengan kode use case UC-03 dapat dilihat pada gambar 3.6.



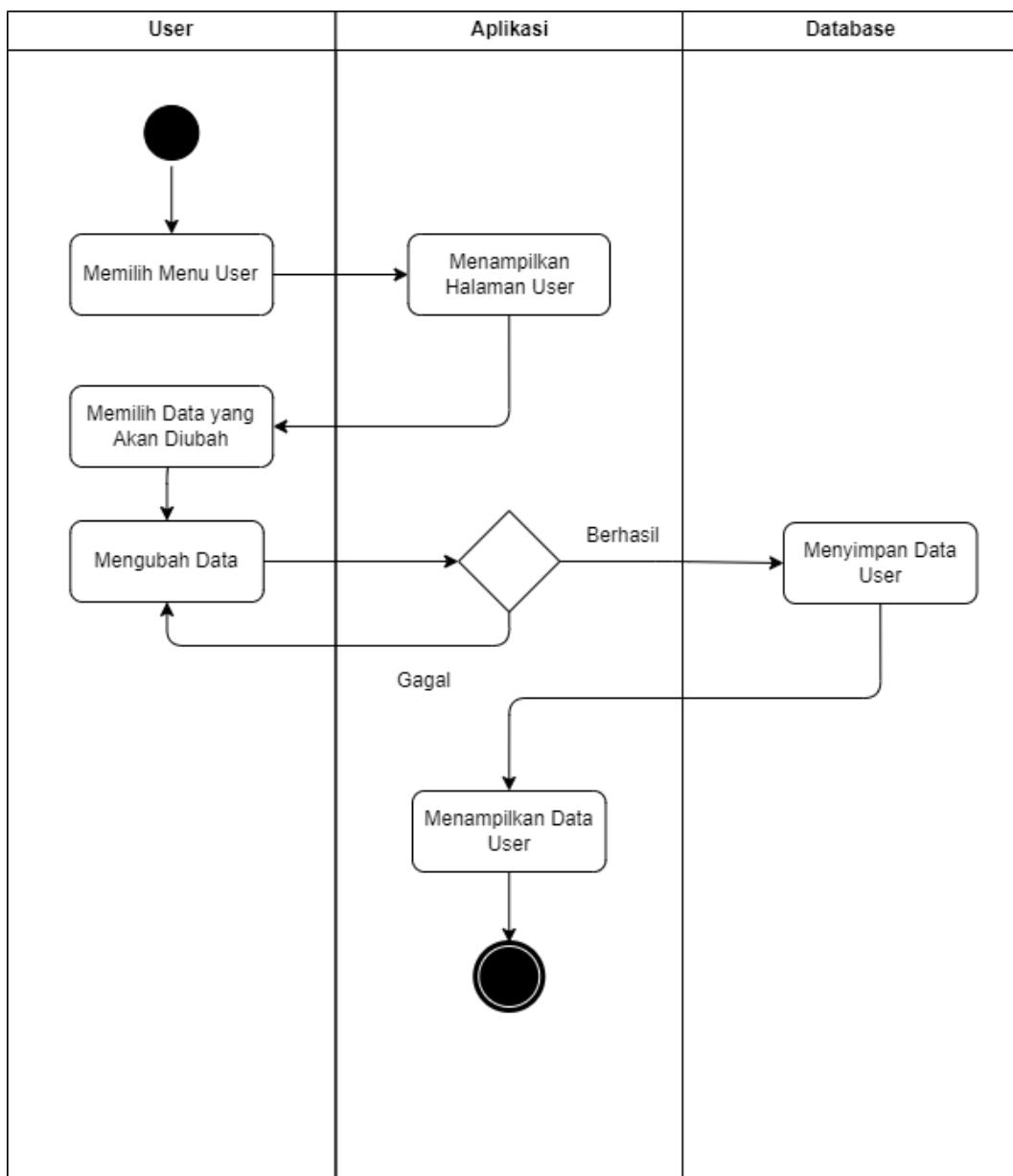
Gambar 3. 6 Activity Diagram UC-03

Untuk activity diagram dengan kode use case UC-04 dapat dilihat pada gambar 3.7.



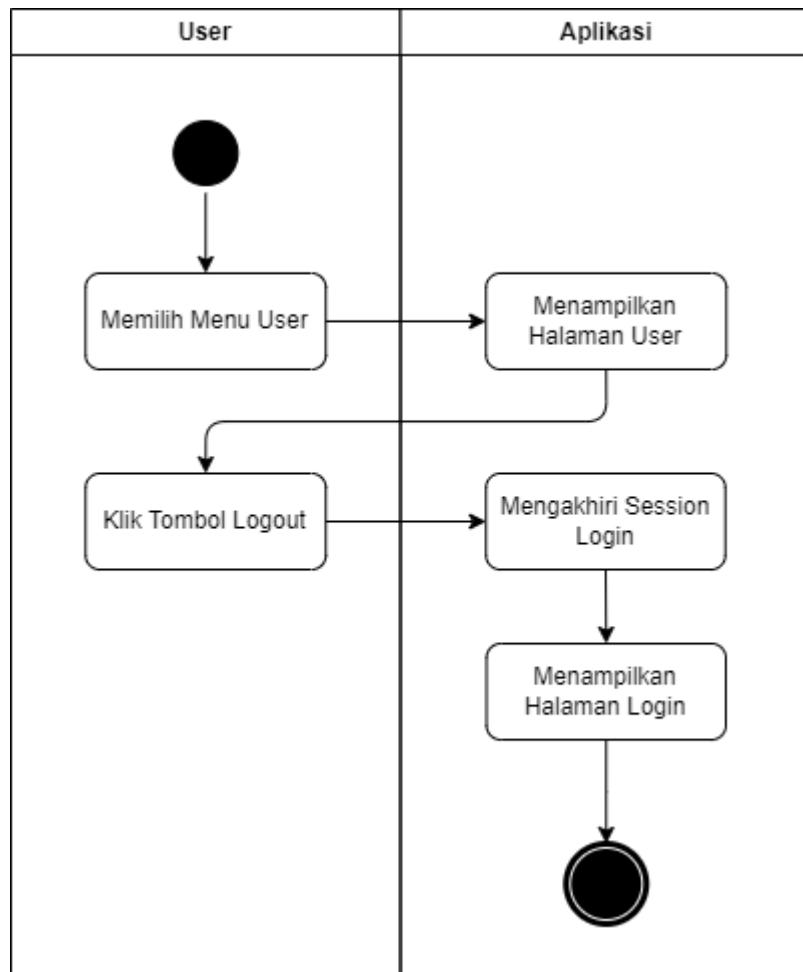
Gambar 3. 7 Activity Diagram UC-04

Untuk activity diagram dengan kode use case UC-05 dapat dilihat pada gambar 3.8.



Gambar 3. 8 Activity Diagram UC-04

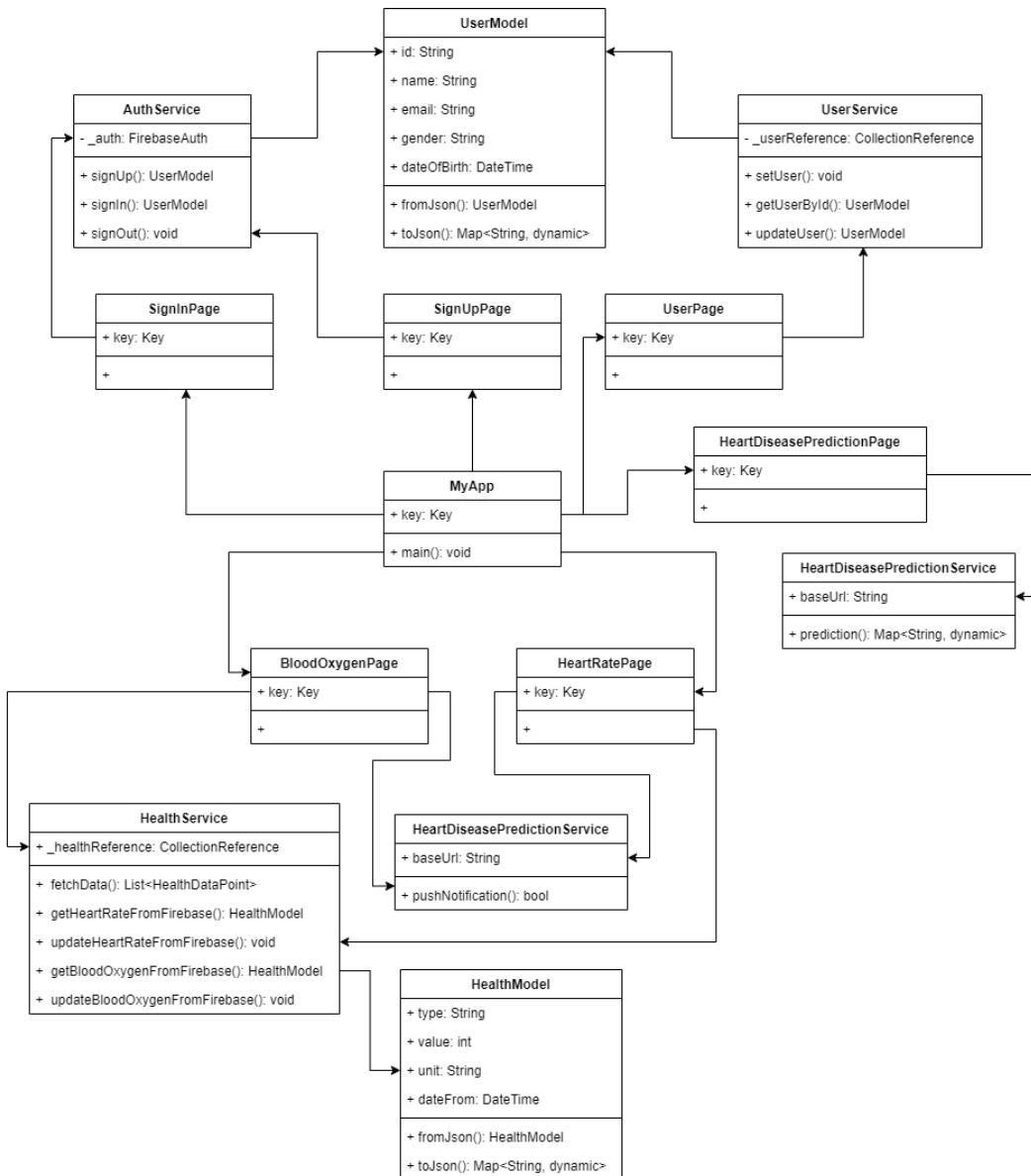
Untuk activity diagram dengan kode use case UC-06 dapat dilihat pada gambar 3.9.



Gambar 3. 9 Activity Diagram UC-06

3.1.7.6. Class Diagram

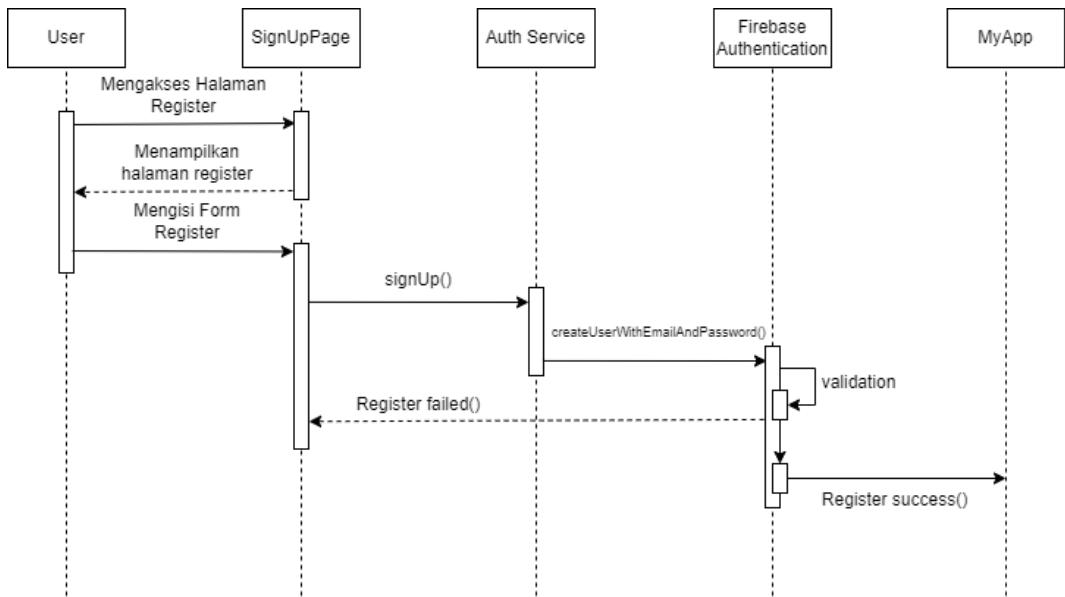
Class diagram adalah penggambaran kelas dari setiap objek yang dibuat dalam aplikasi dan menunjukkan hubungan tiap kelas. Class diagram pada aplikasi ini dapat dilihat pada gambar 3.10



Gambar 3. 10 Class Diagram

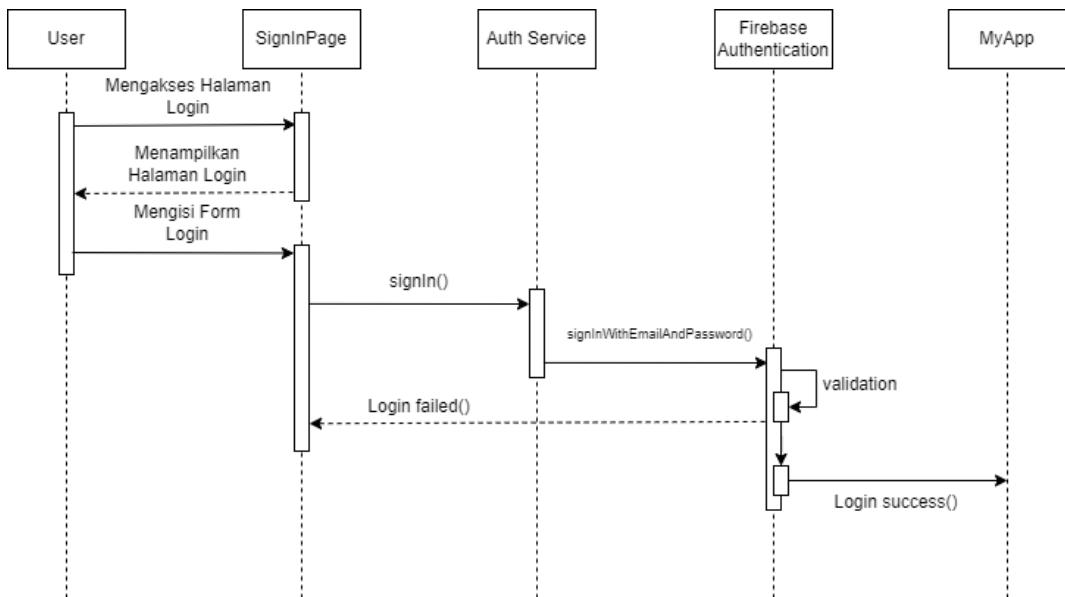
3.1.7.7. Sequence Diagram

Sequence Diagram adalah sebuah diagram yang digunakan untuk menjelaskan dan menampilkan interaksi antar objek-objek dalam sebuah aplikasi secara terperinci. Selain itu sequence diagram juga akan menampilkan pesan atau perintah yang dikirim, beserta waktu pelaksanaanya. Untuk sequence diagram yang pertama, yaitu sequence diagram untuk use case dengan kode UC-01 dapat dilihat pada gambar 3.11.



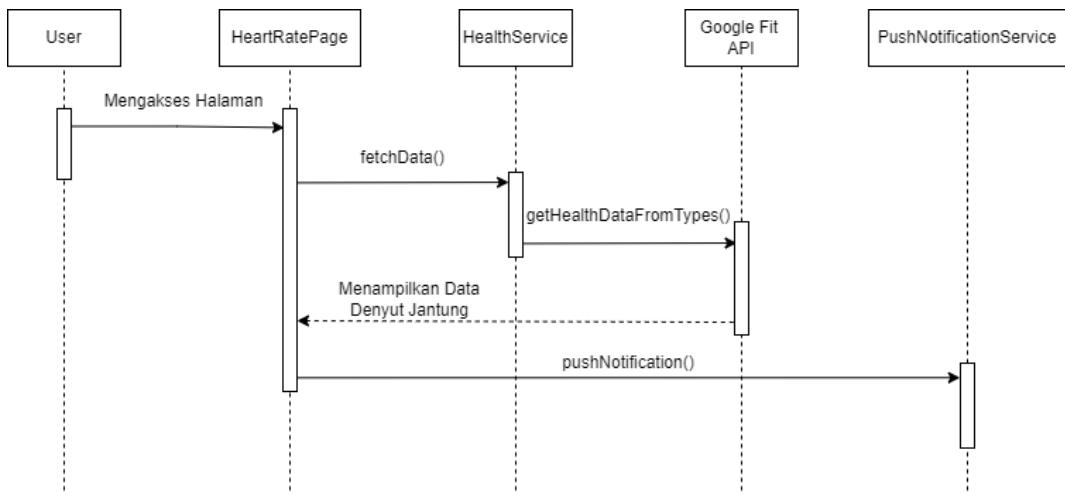
Gambar 3. 11 Sequence Diagram UC-01

Untuk sequence diagram selanjutnya, yaitu sequence diagram untuk use case dengan kode UC-02 dapat dilihat pada gambar 3.12.



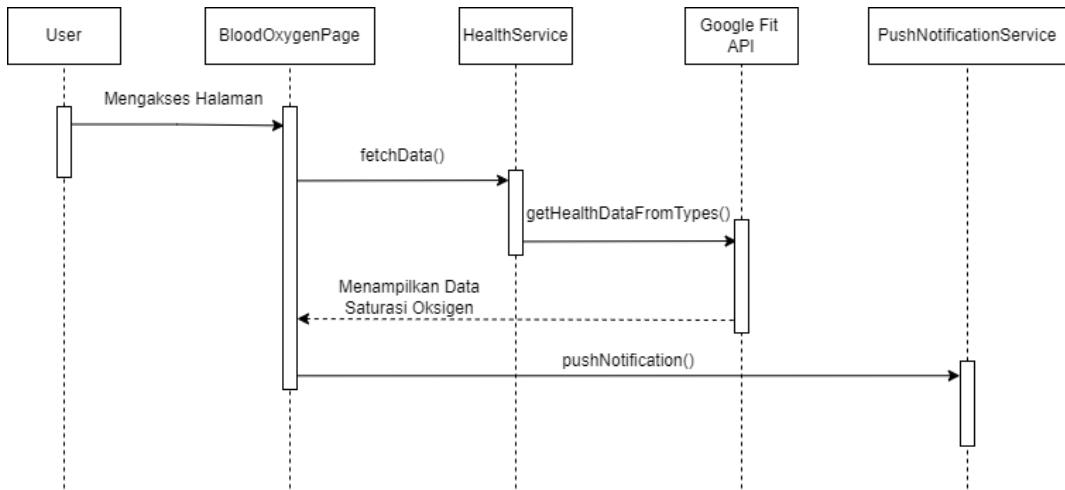
Gambar 3. 12 Sequence Diagram UC-02

Untuk sequence diagram selanjutnya, yaitu sequence diagram untuk use case dengan kode UC-03 dapat dilihat pada gambar 3.13.



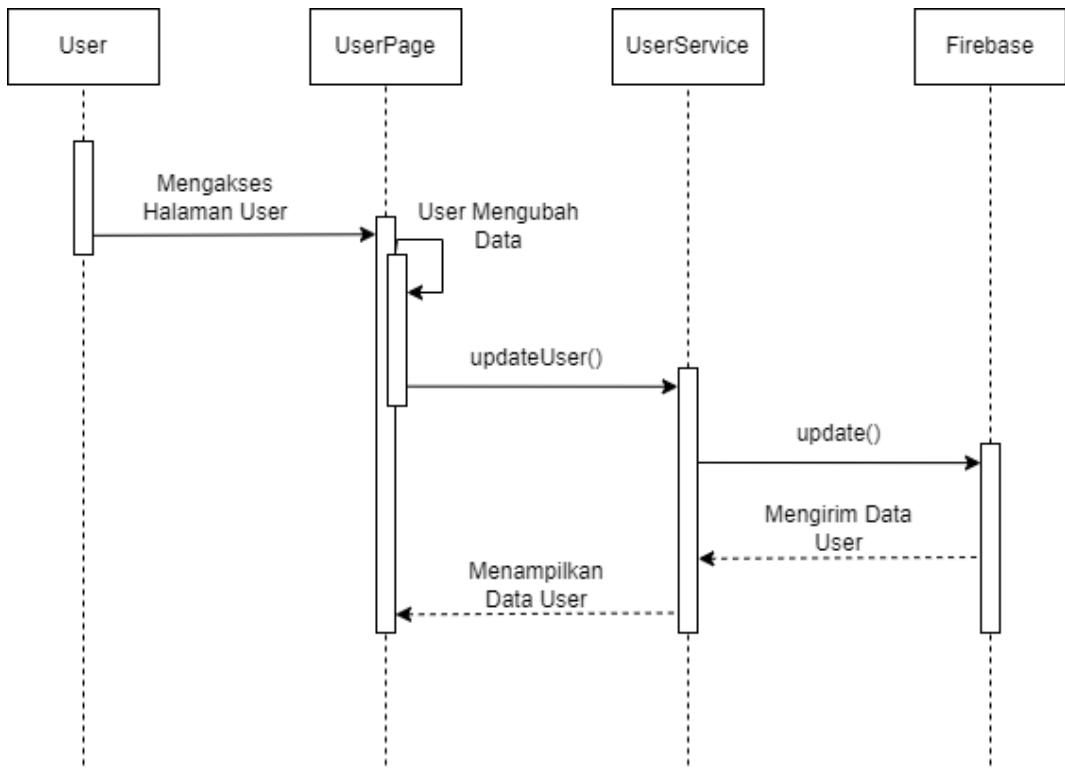
Gambar 3. 13 Sequence Diagram UC-03

Untuk sequence diagram selanjutnya, yaitu sequence diagram untuk use case dengan kode UC-04 dapat dilihat pada gambar 3.14.



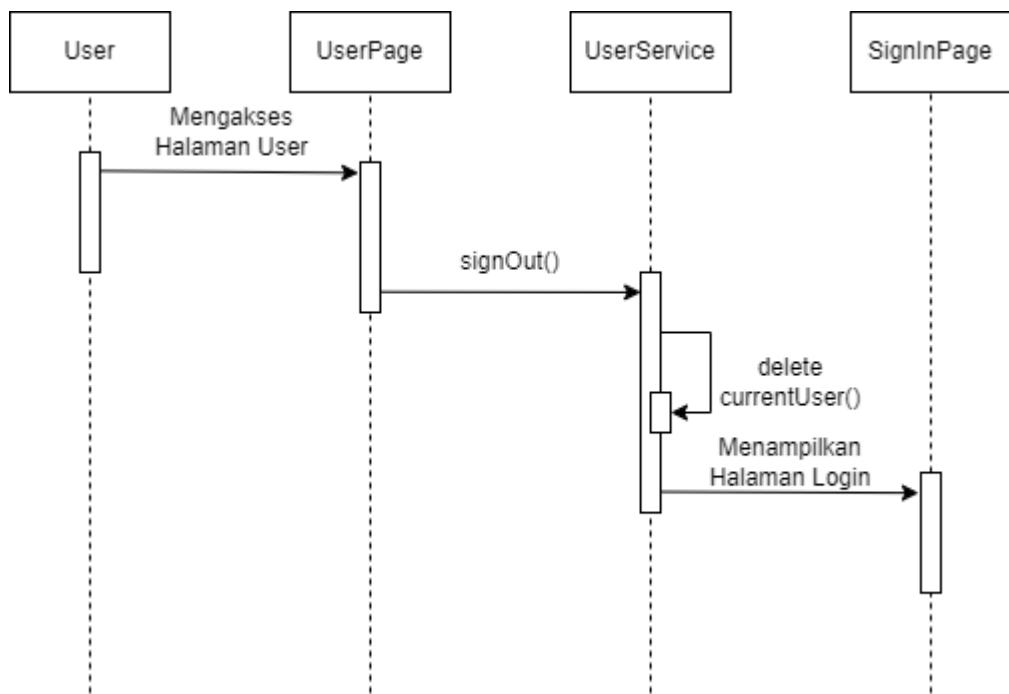
Gambar 3. 14 Sequence Diagram UC-04

Untuk sequence diagram selanjutnya, yaitu sequence diagram untuk use case dengan kode UC-05 dapat dilihat pada gambar 3.15.



Gambar 3. 15 Sequence Diagram UC-05

Untuk sequence diagram selanjutnya, yaitu sequence diagram untuk use case dengan kode UC-06 dapat dilihat pada gambar 3.16.



Gambar 3. 16 Sequence Diagram UC-06

3.2. Perancangan

3.2.1. Perancangan Basis Data

Basis data yang digunakan yaitu Firebase Cloud Firestore. Cloud Firestore ini merupakan database NoSQL yang berorientasi dokumen. Jadi Cloud Firestore ini tidak memiliki tabel atau baris. Sebagai gantinya, data yang disimpan akan berupa dokumen, yang disusun menjadi koleksi. Setiap dokumen akan berisi kumpulan pasangan key dan value.

1. Collection User

Tabel 3. 11 Collection User

Nama Child	Tipe Data
id	String
name	String
email	String
password	String
gender	String

birthDay	Date
----------	------

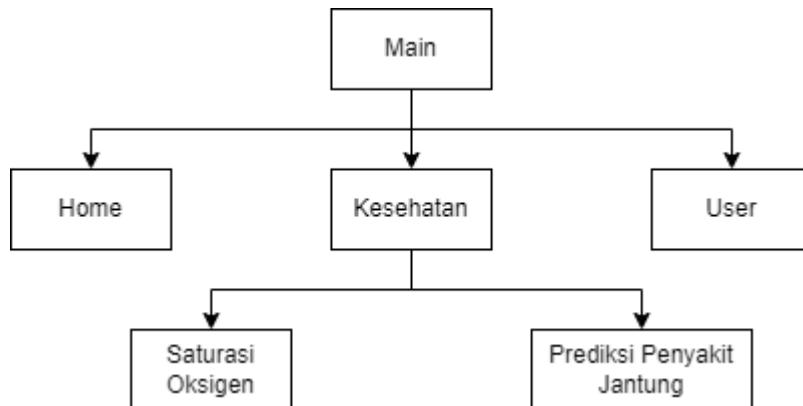
2. Collection Health

Tabel 3. 12 Collection Health

Nama Child	Tipe Data
type	String
value	Number
unit	String
dateFrom	String

3.2.2. Perancangan Struktur Menu

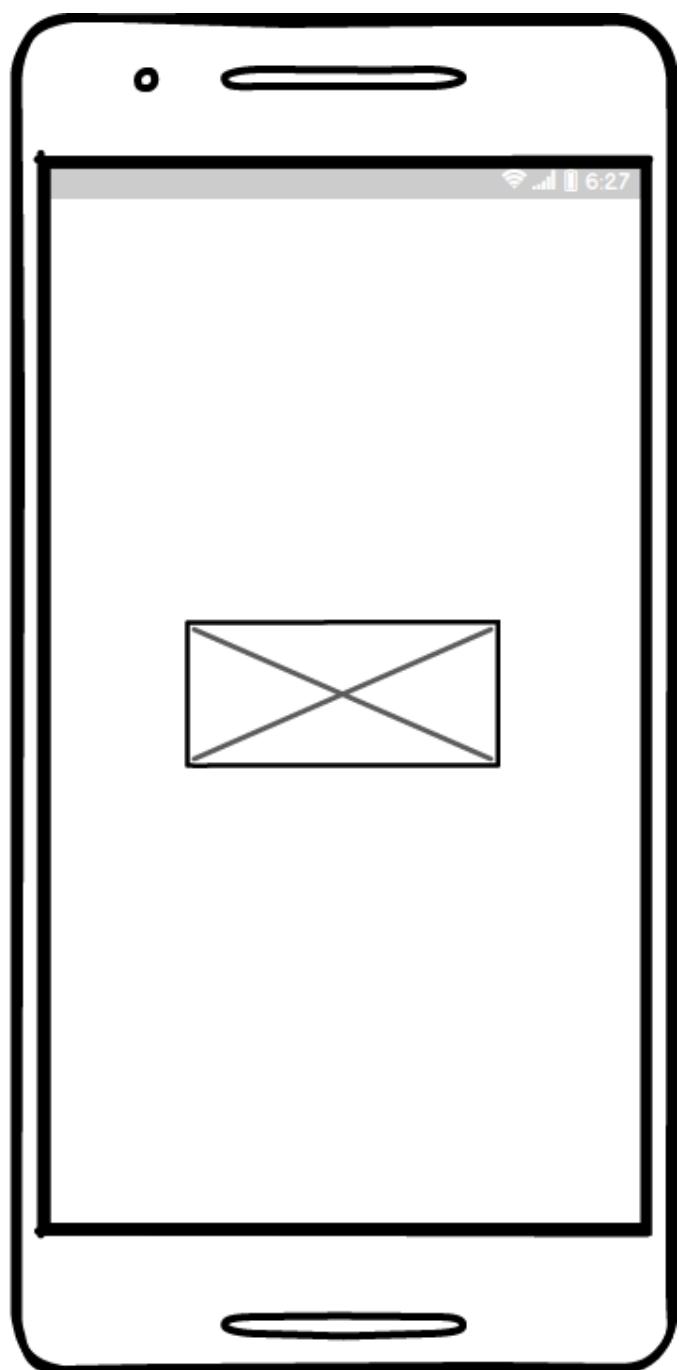
Pembuatan aplikasi monitoring kesehatan memiliki struktur menu yang dapat dilihat pada gambar 3.17.



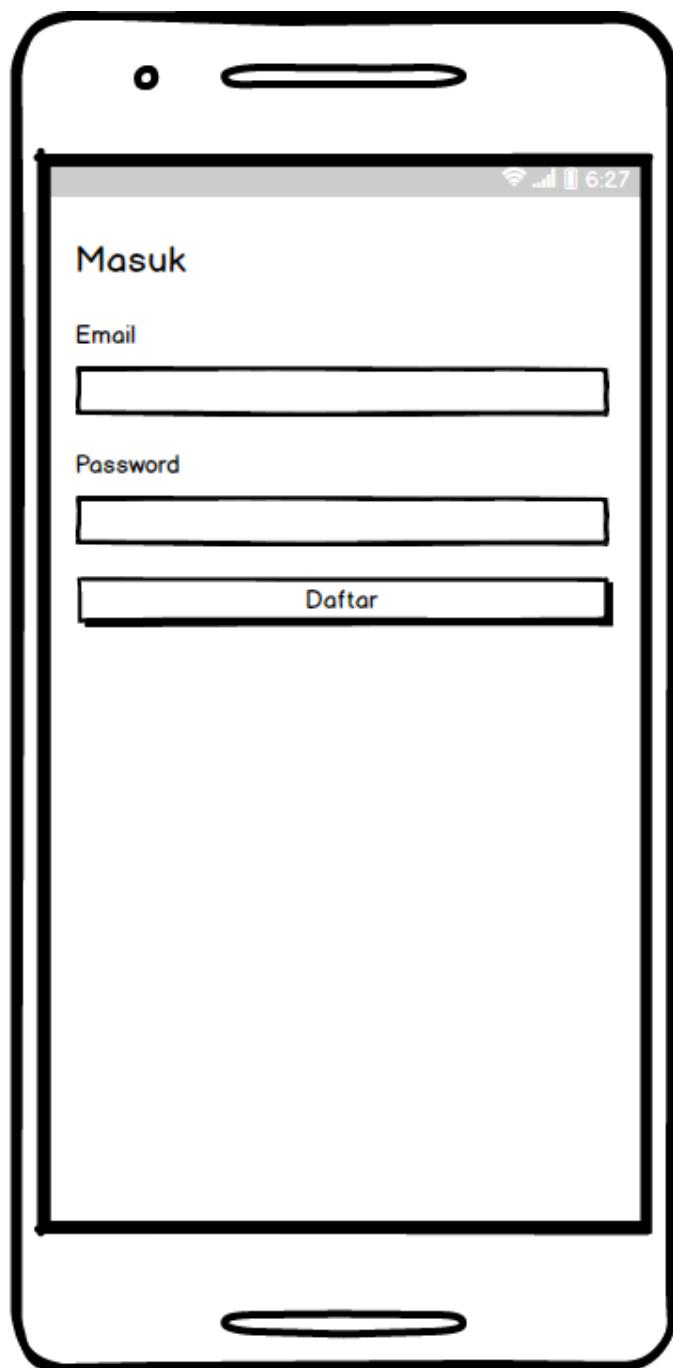
Gambar 3. 17 Struktur Menu

3.2.3. Perancangan Antar Muka

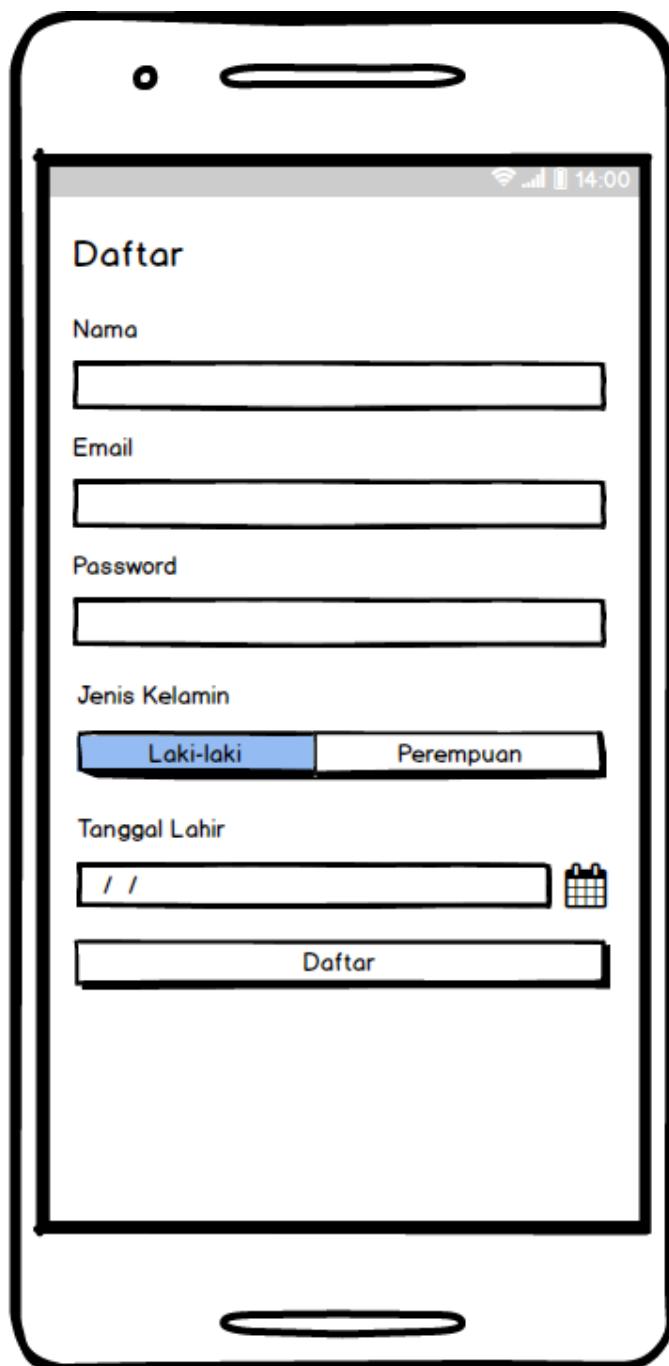
Perancangan antarmuka ini untuk membuat gambar sistem yang akan dibuat nantinya.



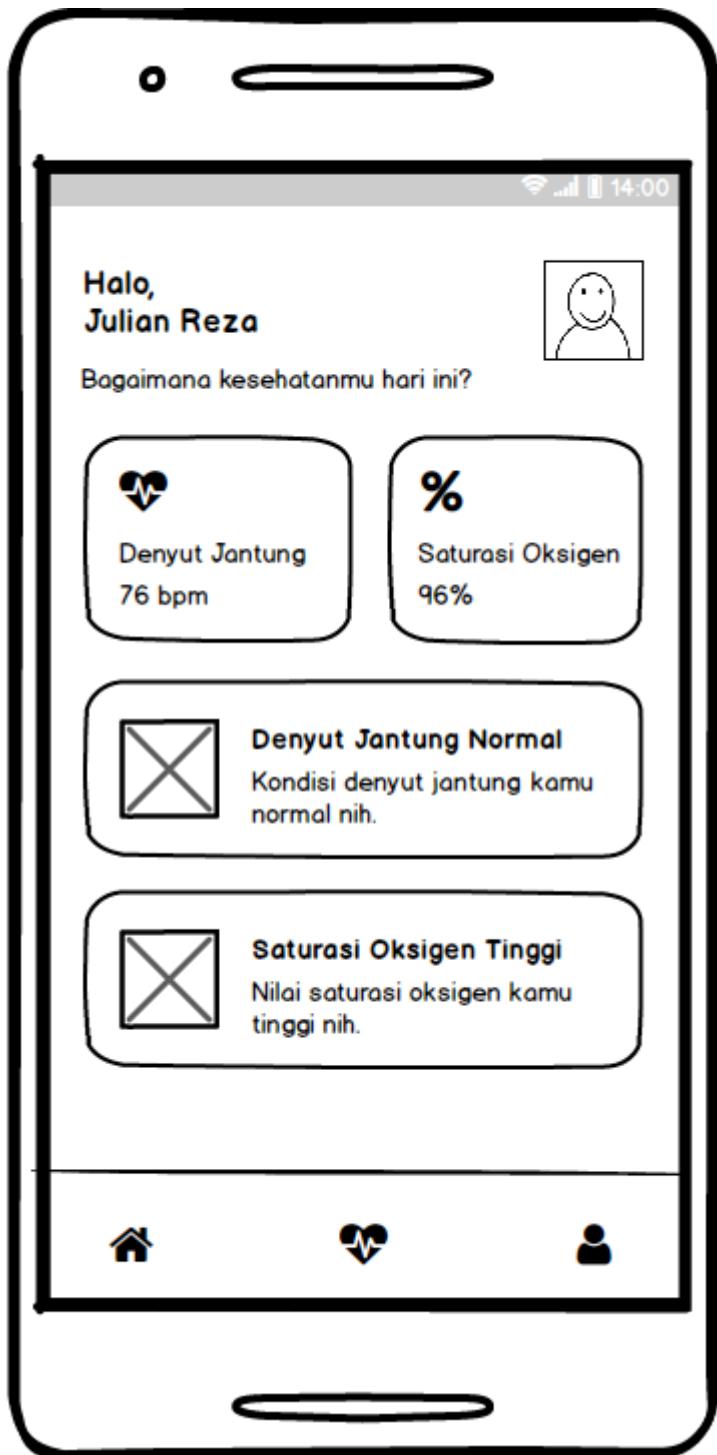
Gambar 3. 18 Rancangan Antarmuka Splash Page



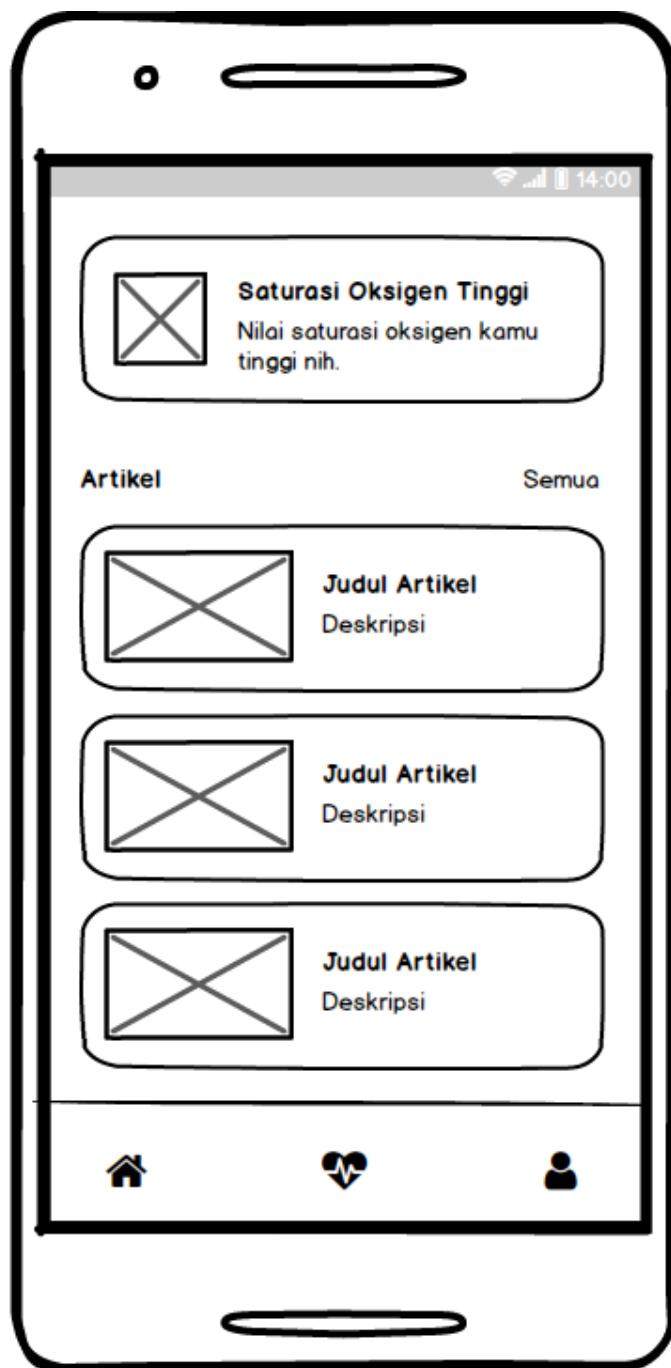
Gambar 3. 19 Rancangan Antarmuka Halaman Login



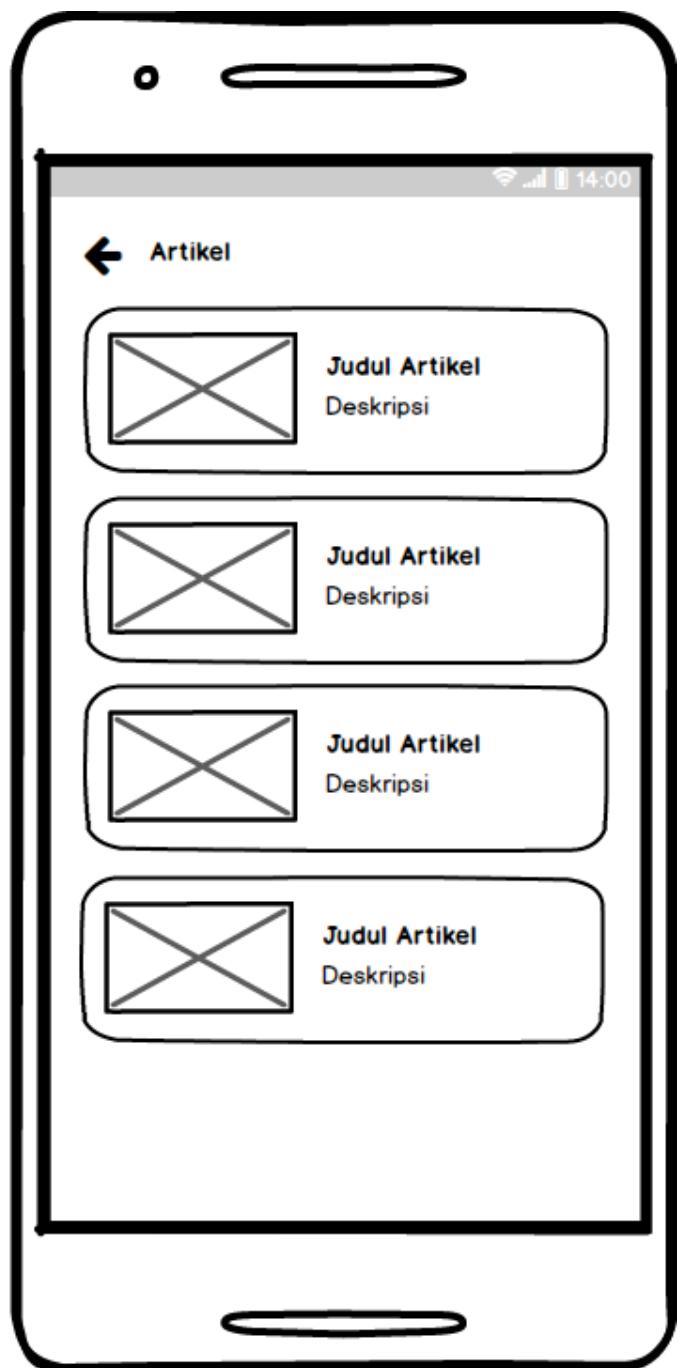
Gambar 3. 20 Rancangan Antarmuka Halaman Register



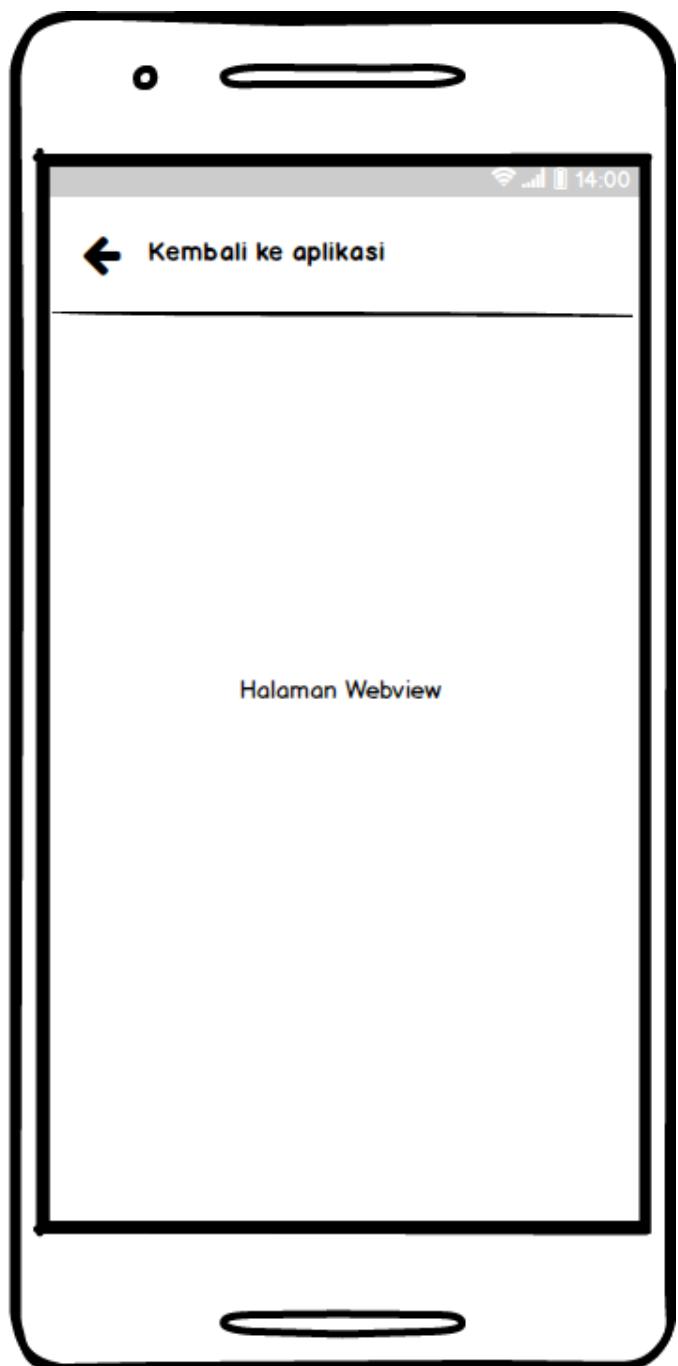
Gambar 3. 21 Rancangan Antarmuka Halaman Home



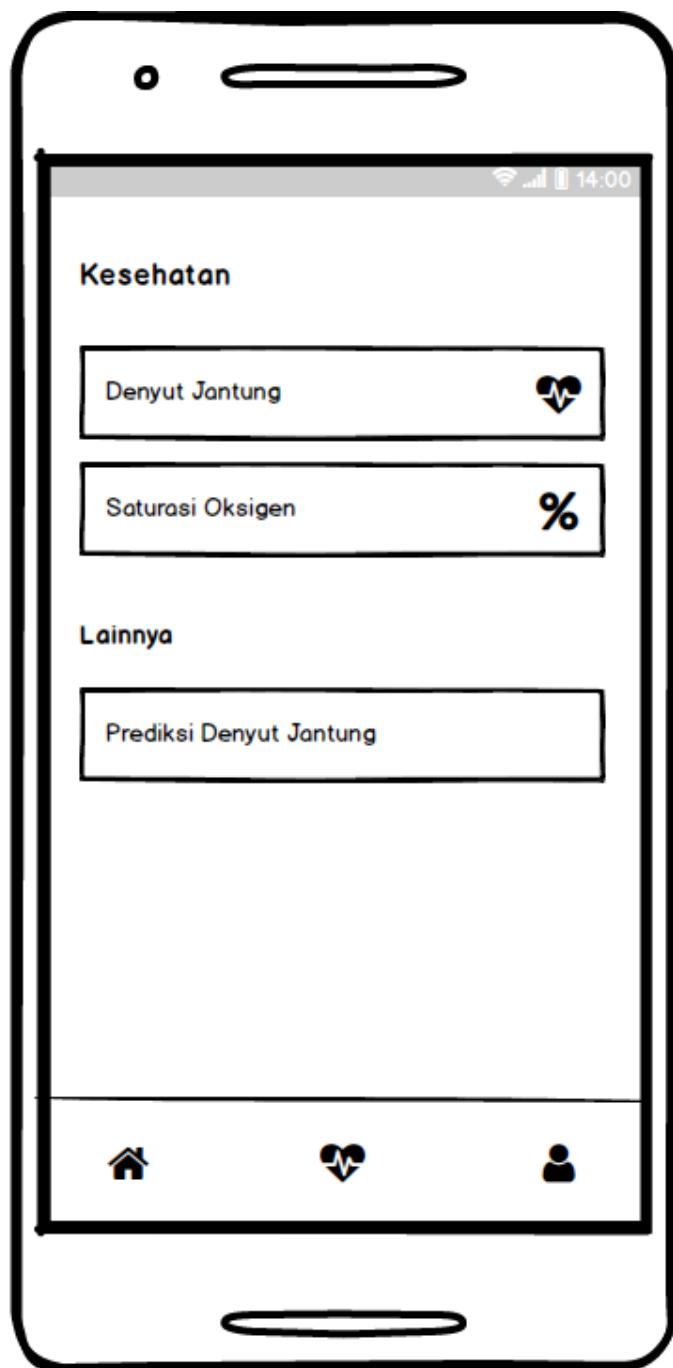
Gambar 3. 22 Rancangan Antarmuka Halaman Home 2



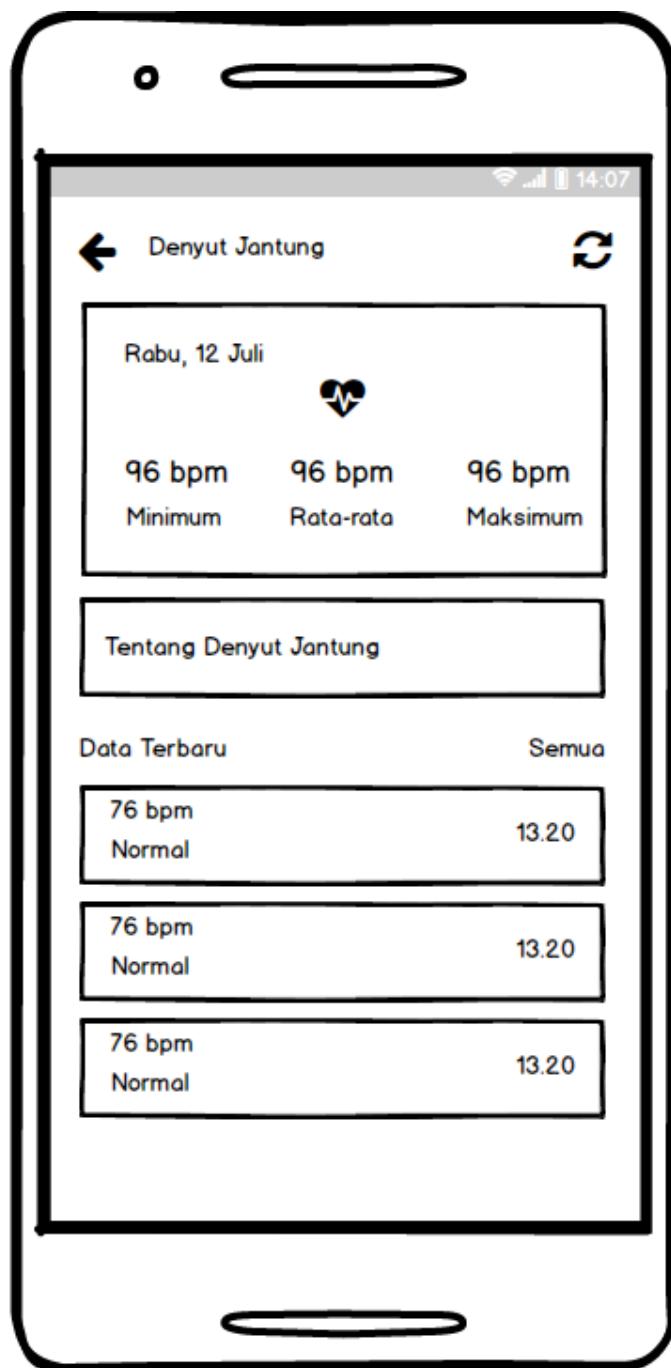
Gambar 3. 23 Rancangan Antarmuka Halaman Semua Artikel



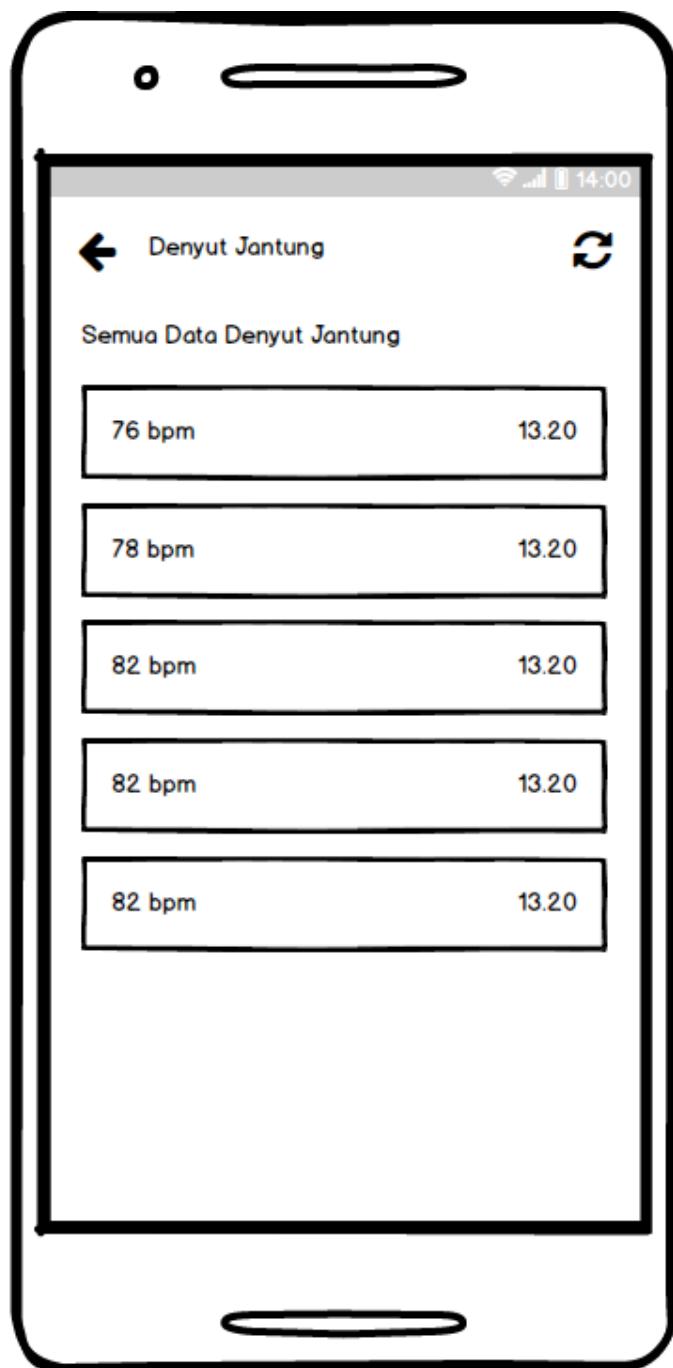
Gambar 3. 24 Rancangan Antarmuka Halaman Artikel Webview



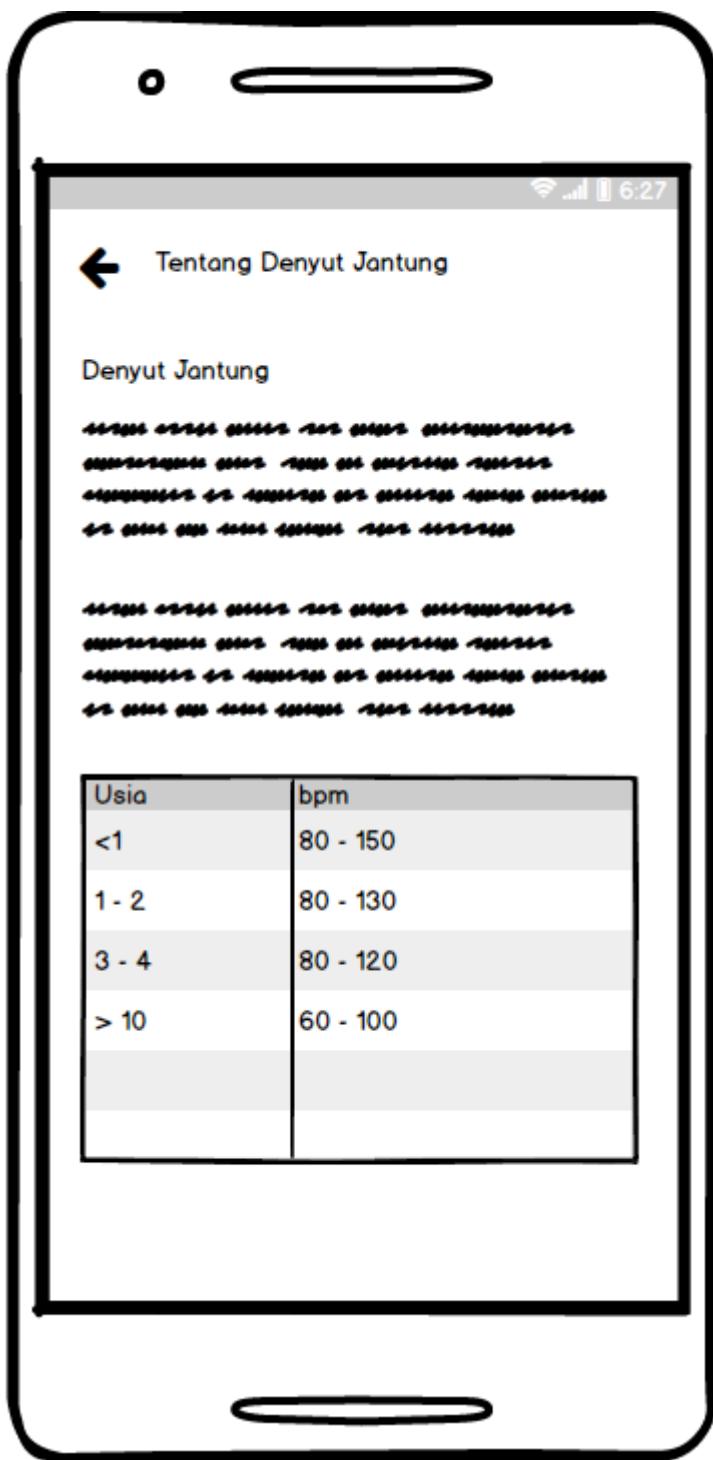
Gambar 3. 25 Rancangan Antarmuka Halaman Kesehatan



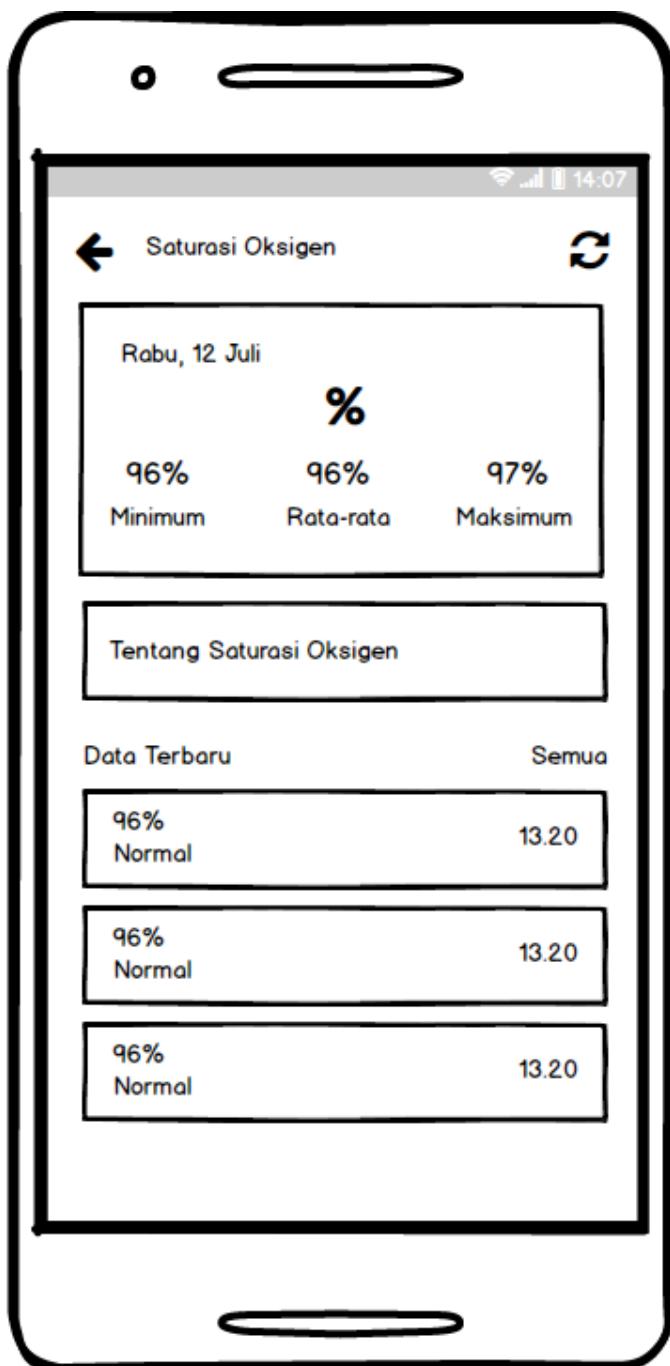
Gambar 3. 26 Rancangan Antarmuka Halaman Denyut Jantung



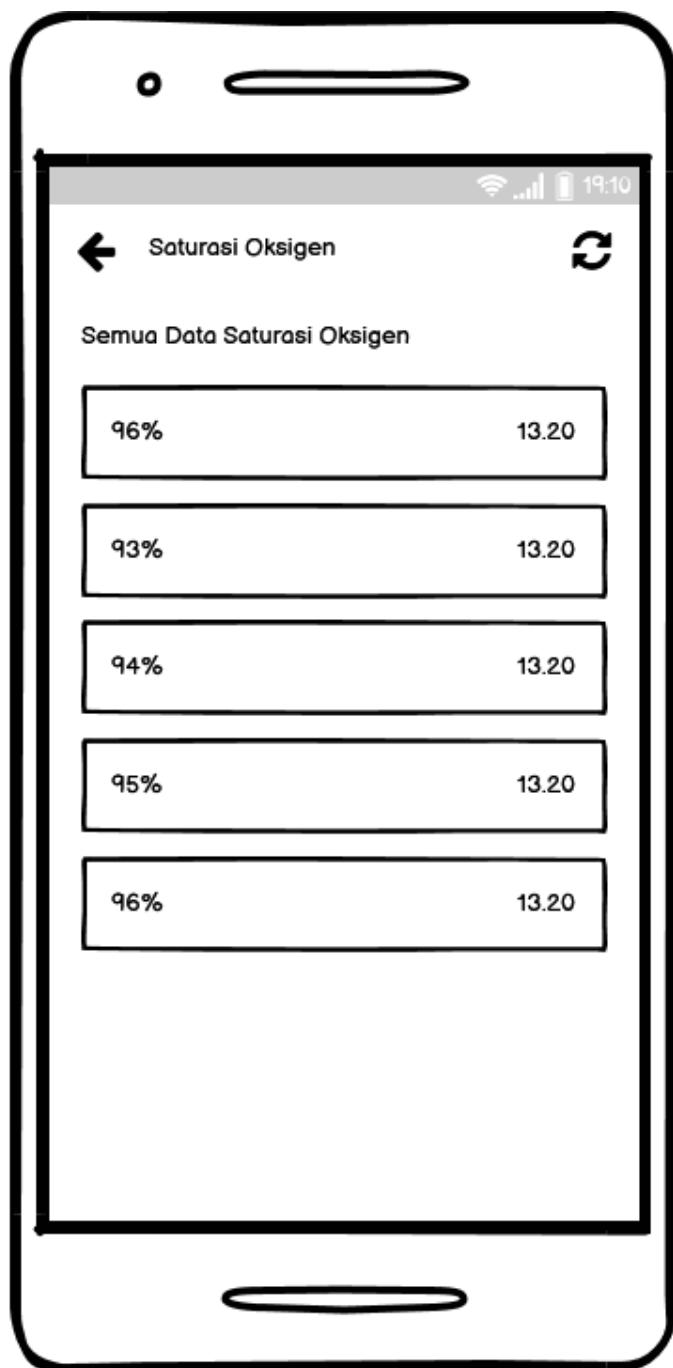
Gambar 3. 27 Rancangan Antarmuka Halaman Data Denyut Jantung



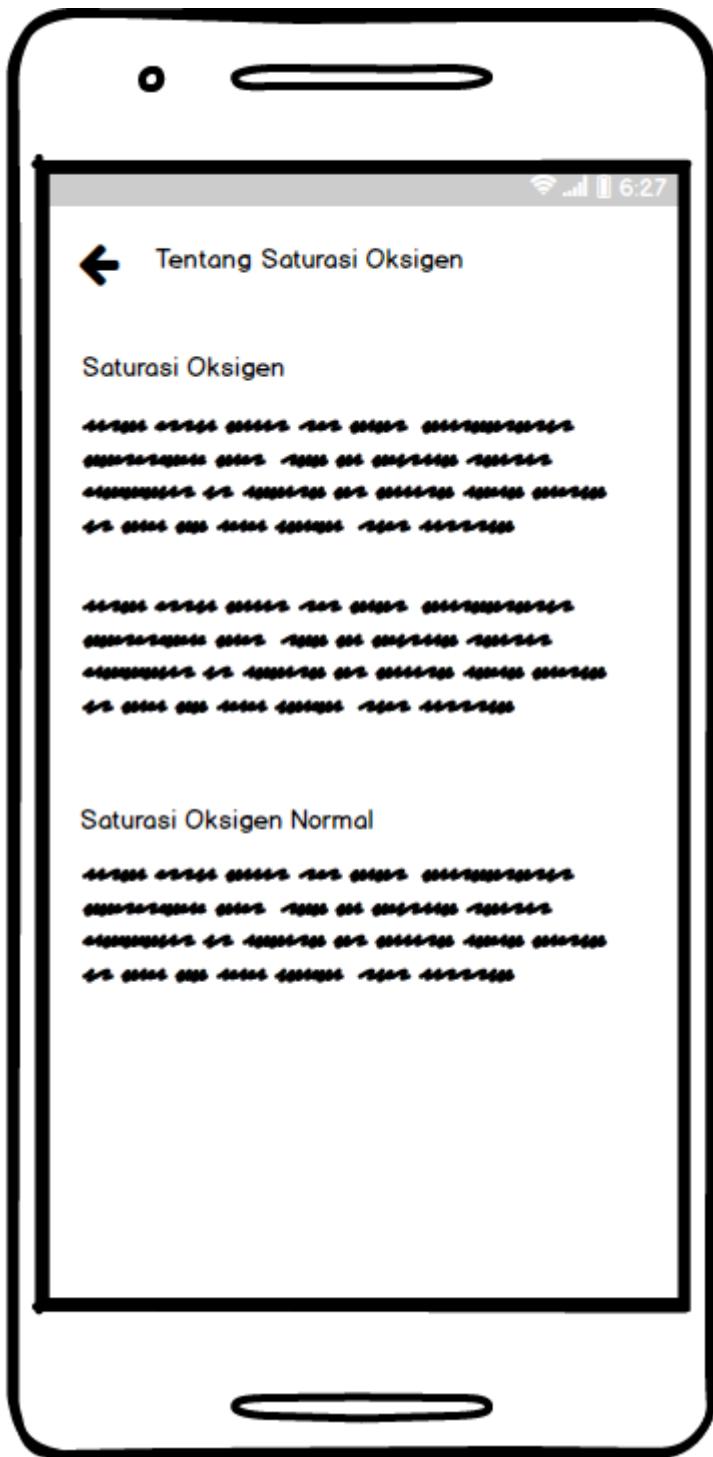
Gambar 3. 28 Rancangan Antarmuka Halaman Tentang Denyut Jantung



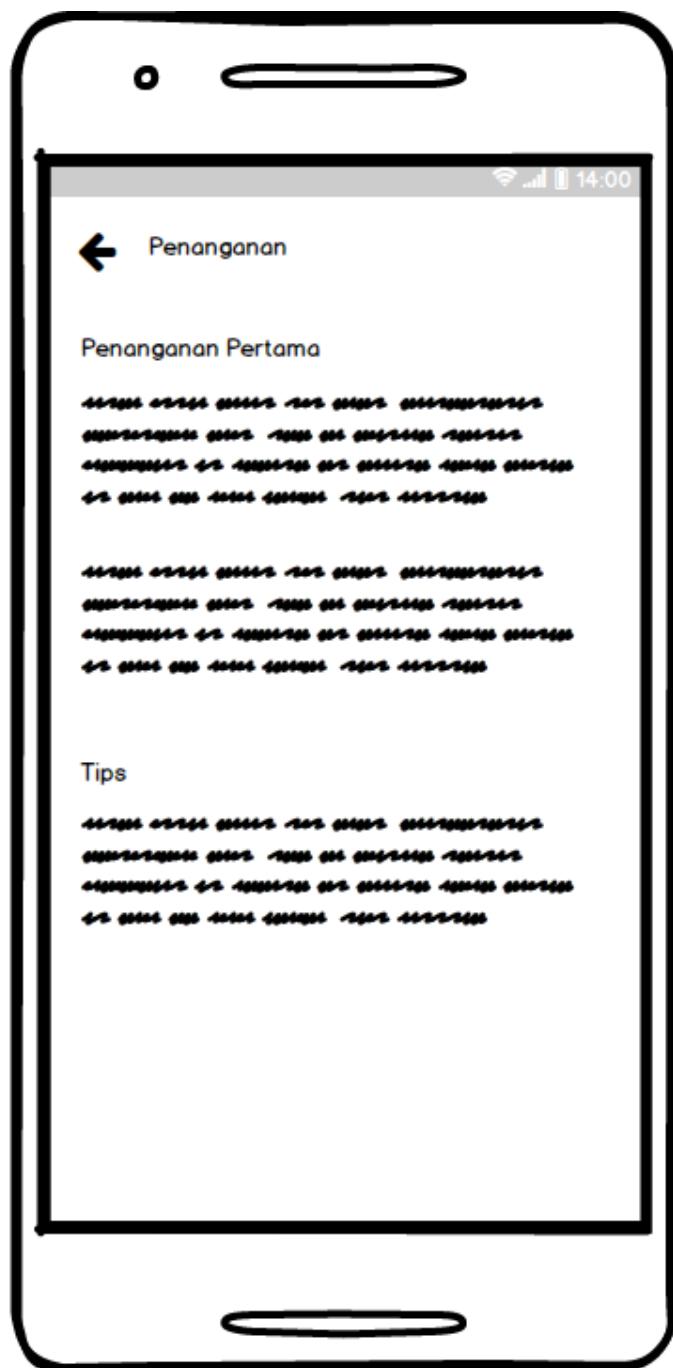
Gambar 3. 29 Rancangan Antarmuka Halaman Saturasi Oksigen



Gambar 3. 30 Rancangan Antarmuka Halaman Data Saturasi Oksigen



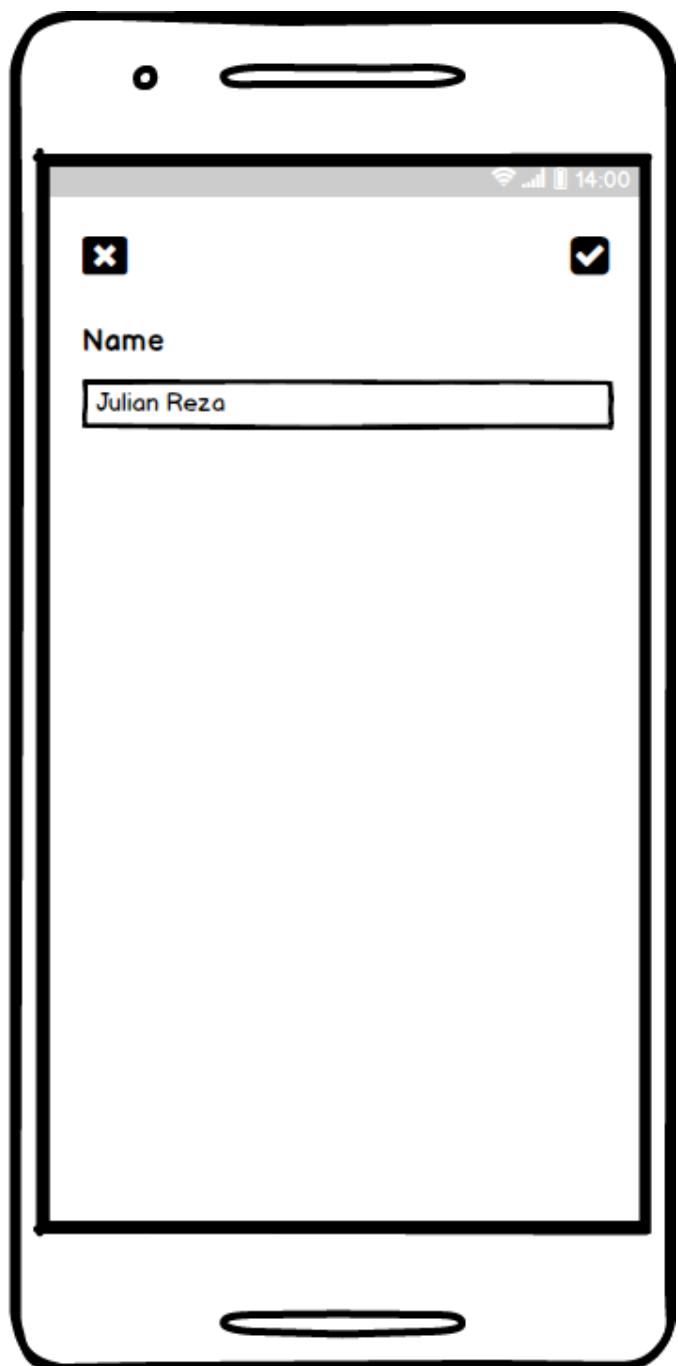
Gambar 3. 31 Rancangan Antarmuka Halaman Tentang Saturasi Oksigen



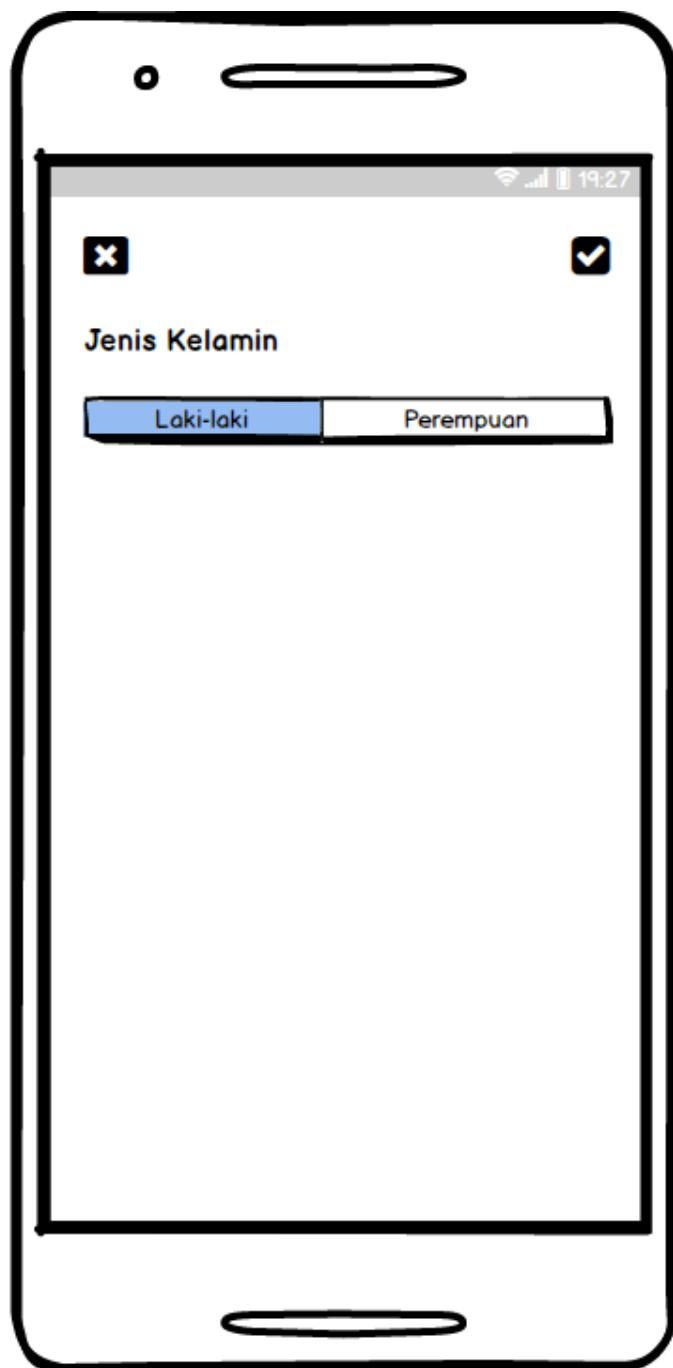
Gambar 3. 32 Rancangan Antarmuka Halaman Penanganan



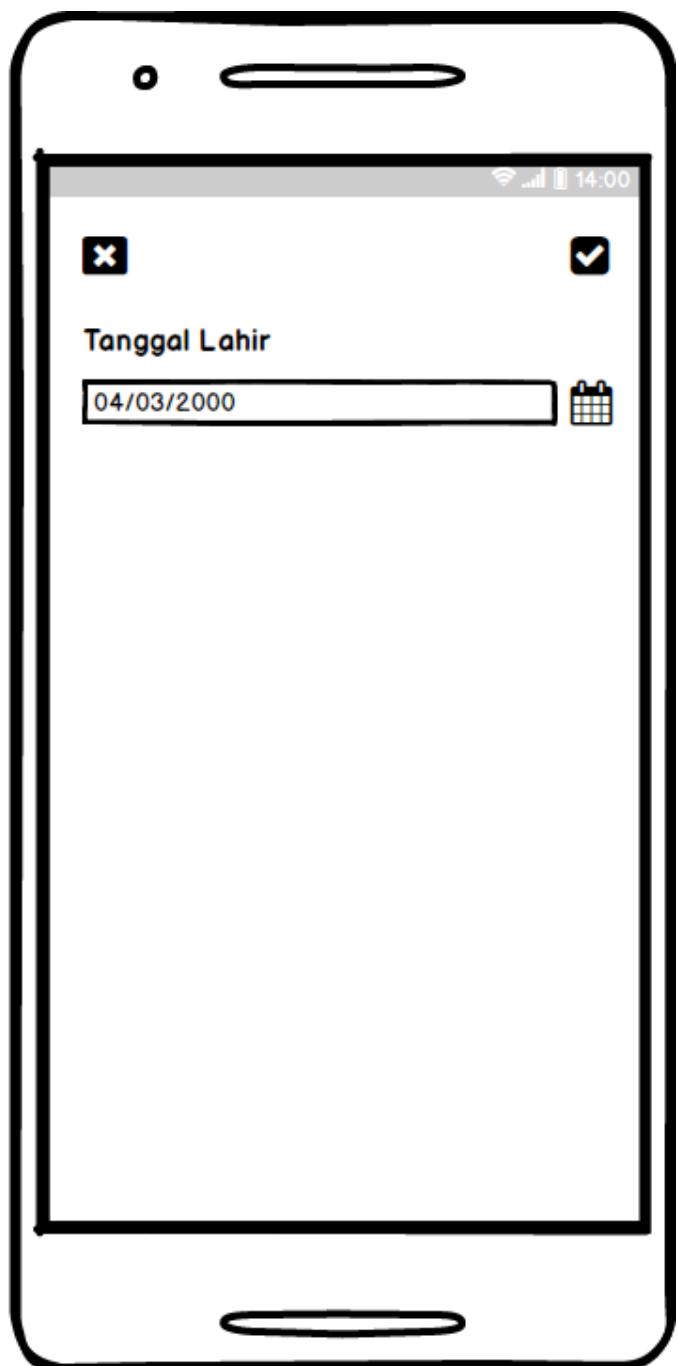
Gambar 3. 33 Rancangan Antarmuka Halaman User



Gambar 3. 34 Rancangan Antarmuka Halaman User Edit Nama



Gambar 3. 35 Rancangan Antarmuka Halaman User Edit Jenis Kelamin



Gambar 3. 36 Rancangan Antarmuka Halaman User Edit Tanggal Lahir

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

Implementasi merupakan tahap menerjemahkan perancangan pada tahap analisis dan perancangan. Tujuan implementasi adalah untuk mengkonfirmasi sistem yang telah dirancang sehingga pengguna dapat memberikan masukan kepada peneliti. Implementasi sistem juga menjelaskan tentang kebutuhan perangkat lunak peneliti, perangkat keras peneliti, implementasi antarmuka, dan implementasi pada basis data.

4.1.1. Implementasi Perangkat Lunak

Untuk perangkat lunak pada laptop yang peneliti gunakan selama melakukan pembangunan dan pengujian sistem, dapat dilihat pada table 4.1 berikut

Tabel 4. 1 Implementasi Perangkat Lunak Laptop

No.	Nama	Spesifikasi
1.	Sistem Operasi	Windows 11 Pro 64 bit
2.	Tools Yang Digunakan	Android Studio Visual Studio Code Postman Genymotion Balsamiq Draw.io Genymobile Scrcpy
3.	Browser	Google Chrome

Untuk perangkat lunak pada smartphone yang peneliti pakai selama melakukan pembangunan dan pengujian sistem, dapat dilihat pada tabel 4.2 berikut

Tabel 4. 2 Implementasi Perangkat Lunak Smartphone

No.	Nama	Spesifikasi
1.	Sistem Operasi	Android 11
2.	UI	MIUI 12.5.5

4.1.2. Implementasi Perangkat Keras

Untuk perangkat keras pada laptop yang peneliti gunakan selama melakukan pembangunan dan pengujian sistem, dapat dilihat pada tabel 4.3 berikut

Tabel 4. 3 Implementasi Perangkat Keras Laptop

No.	Nama	Spesifikasi
1.	CPU	AMD A8
2.	Memory	Dual Channel DDR 3 2x4GB
3.	Storage	128GB ADATA SSD + 500GB HDD
4.	GPU	AMD Radeon R5 Graphics

Sementara untuk perangkat pada smartphone yang peneliti gunakan selama melakukan pembangunan dan pengujian sistem, dapat dilihat pada tabel 4.4 berikut

Tabel 4. 4 Implementasi Perangkat Keras Smartphone

No.	Nama	Spesifikasi
1.	Processor	Mediatek Helio G85
2.	Memory	4GB

3.	Storage	64GB
4.	Display	FHD+
5.	GPU	Mali-G25 MC2

4.1.3. Implementasi Basis Data

Pada implementasi basis data ini dibuat menyesuaikan dengan fungsi yang ada pada aplikasi. Basis data yang digunakan yaitu Firebase, dengan layanan yang dipilih yaitu Cloud Firestore. Cloud Firestore ini merupakan database NoSQL yang berorientasi dokumen. Jadi Cloud Firestore ini tidak memiliki tabel atau baris. Sebagai gantinya, data yang disimpan akan berupa dokumen, yang disusun menjadi koleksi. Setiap dokumen akan berisi kumpulan pasangan key dan value.

1. Collection user

Collection ini digunakan untuk menyimpan data user. Untuk strukturnya sendiri akan berupa pasangan key-value yang meliputi:

```

name: "nama user"
email: "emailuser@gmail.com"
gender: "jenis kelamin user"
birthday: "2000-01-01 00:00:00:000"

```

Untuk password, karena disini kita menggunakan Firebase Authentication dan data tersebut juga merupakan data yang kredensial, jadi untuk data password dan juga email disimpan di dalam Firebase Authentication.

2. Collection health

Collection ini digunakan untuk menyimpan data pengecekan kesehatan yang sudah dilakukan oleh user. Untuk strukturnya sendiri meliputi:

```

type: "tipe dari datanya, contohnya HEART_RATE atau
BLOOD_OXYGEN"
value: "value yang akan berisikan nilai integer, contohnya 93"

```

unit: “satuan unit dari dari tipenya, contohnya PERCENT atau BEATS_PER_MINUTE”

dateFrom: “untuk menyimpan tanggal pengecekan”

Karena basis data yang digunakan yaitu Firebase, atau lebih tepatnya Cloud Firestore, sebenarnya tidak ada query khusus yang digunakan untuk pembuatan setiap dokumennya. Namun pada kasus ini, kita akan membuat sebuah model yang digunakan untuk memudahkan aplikasi untuk berinteraksi dengan basis data.

Tabel 4. 5 Implementasi User Model

Nama Dokumen	Nama Model	Code
user	user_model	<pre>class UserModel { final String id; final String email; final String name; final String gender; final String dateOfBirth; UserModel(required this.id, required this.email, required this.name, required this.gender, required this.dateOfBirth,); factory UserModel.fromJson(String id, Map<String, dynamic> json) => UserModel(id: id,); }</pre>

		<pre> email: json['email'], name: json['name'], gender: json['gender'], dateOfBirth: json['date_of_birth'],); Map<String, dynamic> toJson() => { 'id': id, 'email': email, 'name': name, 'gender': gender, 'date_of_birth': dateOfBirth, }; } </pre>
--	--	---

Tabel 4. 6 Implementasi Health Model

Nama Dokumen	Nama Model	Code
health	health_model	<pre> class HealthModel { final String type; final int value; final String unit; final String dateFrom; HealthModel({ required this.type, required this.value, required this.unit, required this.dateFrom, }); } </pre>

		<pre> factory HealthModel.fromJson(Map<String, dynamic> json) => HealthModel(type: json['type'], value: json['value'], unit: json['unit'], dateFrom: json['dateFrom'],); Map<String, dynamic> toJson() => { 'type': type, 'value': value, 'unit': unit, 'dateFrom': dateFrom, }; } </pre>
--	--	--

4.1.4. Implementasi Antarmuka

Implementasi antarmuka menjelaskan tentang implementasi antarmuka sistem pada aplikasi android. Implementasi antarmuka diwakili dengan nama antarmuka beserta nama filenya. Implementasi antarmuka sistem dapat dilihat pada tabel 4.7 berikut.

Tabel 4. 7 Implementasi Antarmuka

No.	Nama Antarmuka	Nama File
1	Halaman Splash Page	splash_page.dart
2	Halaman Login	sign_in_page.dart
3	Halaman Register	sign_up_page.dart

4	Halaman Home	home_page.dart
5	Halaman Semua Artikel	article_page.dart
6	Halaman Artikel Webview	webview_page.dart
7	Halaman Kesehatan	health_page.dart
8	Halaman Denyut Jantung	heart_rate_page.dart
9	Halaman Data Denyut Jantung	all_heart_rate_page.dart
10	Halaman Tentang Denyut Jantung	heart_rate_information_page.dart
11	Halaman Saturasi Oksigen	blood_oxygen_page.dart
12	Halaman Data Saturasi Oksigen	all_blood_oxygen_page.dart
13	Halaman Tentang Saturasi Oksigen	blood_oxygen_information_page.dart
14	Halaman Penanganan Denyut Jantung Rendah	heart_rate_low_page.dart
15	Halaman Penanganan Denyut Jantung Tinggi	heart_rate_high_page.dart
16	Halaman Penanganan Saturasi Oksigen Rendah	blood_oxygen_low_handling_page.dart
17	Halaman User	user_page.dart
18	Halaman User Edit Nama	edit_name_page.dart
19	Halaman User Edit Jenis Kelamin	edit_gender_page.dart
20	Halaman User Edit Tanggal Lahir	edit_date_of_birth_page.dart

4.1.5. Implementasi Teknologi

Implementasi teknologi adalah pembahasan hasil dari beberapa teknologi yang digunakan dalam pembangunan sistem. Adapun teknologi yang digunakan beserta implementasinya adalah sebagai berikut.

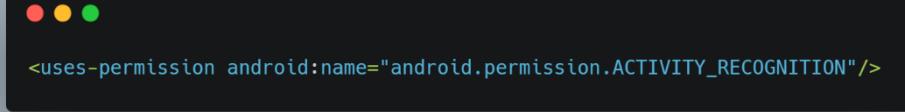
4.1.5.1. Implementasi Google Fit API

Implementasi Google Fit API pada pembangunan aplikasi ini agar aplikasi dapat mengambil data seperti data denyut jantung dan saturasi oksigen ke aplikasi Google Fit. Karena pada membuat aplikasi ini kita menggunakan flutter sebagai teknologi utama, maka ada sebuah package yang dapat membantu kita untuk menghubungkan aplikasi dengan Google Fit dengan cukup mudah, yaitu dengan package health. Langkah pertama dalam mengimplementasikan Google Fit API yaitu dengan menginstall atau menambahkan package health ke aplikasi kita dengan syntax berikut:

```
$ flutter pub add health
```

Gambar 4. 1 Menambahkan Package ke Dalam Aplikasi

Selanjutnya yaitu menambahkan permission baru pada file manifest.



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

Gambar 4. 2 Menambahkan permission baru ke manifest

Setelah semua langkah diatas dilakukan, maka langkah terakhir adalah mengimplementasikan kode untuk Google Fit API ke dalam file dart kita. Untuk detailnya bisa dilihat pada skrip berikut.

Skrip 4. 1 Kelas untuk penggunaan Google Fit API

```
import 'package:health/health.dart';

class HealthService {

    Future<List<HealthDataPoint>> fetchData(
        DateTime now,
        DateTime yesterday,
        List<HealthDataType> types,
    ) async {
        List<HealthDataPoint> healthDataList = [];
        HealthFactory health = HealthFactory();

        final permissions = [
            HealthDataAccess.READ,
        ];

        bool requested =
            await health.requestAuthorization(types, permissions:
permissions);

        await Permission.activityRecognition.request();
        await Permission.location.request();

        if (requested) {
            try {
                List<HealthDataPoint> healthData =
                    await health.getHealthDataFromTypes(yesterday, now,
types);
                healthDataList.addAll((healthData.length < 100)
                    ? healthData
                    : healthData.sublist(0, 100));
            } catch (error) {
                throw Exception("Exception in getHealthDataFromTypes:
$error");
            }
        }

        healthDataList =
```

```

    HealthFactory.removeDuplicates(healthDataList);

    healthDataList.sort((a, b) {
        var aDate = a.dateFrom;
        var bDate = b.dateFrom;
        return bDate.compareTo(aDate);
    });

    return healthDataList;
} else {
    throw Exception("Authorization not granted");
}
}
}
}

```

4.2. Pengujian

Tahapan penujian sistem ini dilakukan untuk memastikan apakah semua fungsi di dalam sistem dapat berjalan dengan baik sesuai dengan spesifikasi kebutuhan dan mencari kesalahan yang mungkin bisa terjadi pada sistem. Pengujian yang dilakukan pada sistem adalah pengujian alpha secara fungsional. Metode pengujian yang digunakan adalah metode pengujian blackbox yang berfokus pada persyaratan fungsional dari sistem.

4.2.1. Rencana Pengujian Alpha

Untuk rencana pengujian pada sistem yang dibangun, dapat dilihat pada tabel 4.7.

Tabel 4. 8 Rencana Pengujian Alpha

No.	Fungsional yang Diuji	Detail Pengujian	Pengujian
1	Login	User memasukan email dan password yang digunakan untuk masuk ke dalam aplikasi	Blackbox
2	Register	User memasukan nama, email,	Blackbox

		password, jenis kelamin dan tanggal lahir.	
3	Deteksi Denyut Jantung	Sistem menganalisis data denyut jantung untuk mendapat informasi lebih lanjut	Blackbox
4	Deteksi Saturasi Oksigen	Sistem menganalisis data saturasi oksigen untuk mendapat informasi lebih lanjut	Blackbox
5	Edit Profil	User memperbarui data profil mereka apabila ada kesalahan pada saat proses registrasi	Blackbox

4.2.2. Hasil Pengujian

Kasus dan Hasil pengujian merupakan tahapan untuk mengetahui apakah fungsionalitas dari sistem yang telah dibangun sesuai dengan kebutuhan atau belum. Hasil pengujian pada sistem dapat dilihat sebagai berikut

1. Pengujian Login

Pada kasus login, user harus menggunakan akun yang sebelumnya sudah terdaftar pada aplikasi untuk bisa masuk ke dalam sistem. Hasil dari pengujian login ini dapat dilihat pada tabel 4.8.

Tabel 4. 9 Hasil Pengujian Kasus Login

Kasus dan Hasil Uji (Data Benar)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Email: julianreza@gmail.com Password: julianreza	User berhasil login dan mengarahkan user ke halaman	User berhasil login dan langsung diarahkan ke	Diterima

	utama aplikasi.	halaman utama aplikasi.	
Kasus dan Hasil Uji (Data Kosong)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Email: Password:	Menampilkan pesan bahwa email dan juga password tidak boleh kosong		
Kasus dan Hasil Uji (Data Salah)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Email: julianreza@gmail.com Password: juliannn	Menampilkan pesan error		

2. Pengujian Register

Pada kasus ini, user yang akan masuk ke dalam sistem harus mendaftar terlebih dahulu dengan melengkapi data-data sudah disediakan. Untuk hasil pengujinya dapat dilihat pada tabel 4.9.

Tabel 4. 10 Hasil Pengujian Kasus Register

Kasus dan Hasil Uji (Data Benar)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Nama: Julian Reza Email: julianreza@gmail.com Password: julianreza Jenis Kelamin: Laki-laki Tanggal Lahir: 2000-07-01 00:00:00.000	User berhasil register dan mengarahkan user ke halaman utama aplikasi.	User berhasil register dan langsung diarahkan ke halaman utama aplikasi.	Diterima

Kasus dan Hasil Uji (Data Kosong)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Nama Email: Password: Jenis Kelamin: Tanggal Lahir:	Menampilkan pesan bahwa email dan juga password tidak boleh kosong	Sistem menampilkan pesan error bahwa data yang dimasukkan kosong	Diterima
Kasus dan Hasil Uji (Data Salah)			
Nama: Julian Reza Email: julian Password: juliannn Jenis Kelamin: Laki-laki Tanggal Lahir: 2002-08-01 00:00:00.000	Menampilkan pesan error	Sistem menampilkan pesan error bahwa format email yang dimasukkan tidak valid	Diterima

3. Pengujian Deteksi Denyut Jantung

Pada kasus pengujian ini, sistem akan mengambil data denyut jantung dari Google Fit yang didapatkan dari smartwatch untuk selanjutnya diolah di dalam aplikasi. Untuk hasil pengujianya, dapat dilihat pada tabel 4.10.

Tabel 4. 11 Hasil Pengujian Deteksi Denyut Jantung

Kasus dan Hasil Uji (Data Benar)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Sistem: Terhubung dengan Google Fit Smartwatch: Digunakan oleh user.	Sistem berhasil mengambil data denyut jantung dari Google Fit	Sistem berhasil mengambil data denyut jantung, lalu	Diterima

	yang diperoleh dari smartwatch, memproses datanya, menampilkan hasilnya kepada user, dan mengirimkan notifikasi ke smartwatch jika terdeteksi data berada diluar normal.	mengklasifikasi kan datanya, menampilkan hasilnya kepada user, dan mengirimkan notifikasi ke smartwatch jika terdeteksi data berada diluar normal.	
--	--	--	--

4. Pengujian Deteksi Saturasi Oksigen

Pada kasus pengujian ini, sistem akan mengambil data saturasi oksigen dari Google Fit yang didapatkan dari smartwatch untuk selanjutnya diolah di dalam aplikasi. Untuk hasil pengujianya, dapat dilihat pada tabel 4.11.

Tabel 4. 12 Hasil Pengujian Deteksi Saturasi Oksigen

Kasus dan Hasil Uji (Data Benar)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Sistem: Terhubung dengan Google Fit Smartwatch: Digunakan oleh user.	Sistem berhasil mengambil data saturasi oksigen dari Google Fit yang diperoleh dari smartwatch. memproses datanya, menampilkan	Sistem berhasil mengambil data saturasi oksigen, lalu mengklasifikasi kan datanya, menampilkan hasilnya kepada user, dan	Diterima

	hasilnya kepada user, dan mengirimkan notifikasi ke smartwatch jika terdeteksi data berada diluar normal.	mengirimkan notifikasi ke smartwatch jika terdeteksi data berada diluar normal.	
--	---	---	--

5. Pengujian Edit Profil

Jika pada saat proses registrasi ada kesalahan pada data masukkan, maka user dapat memperbarui data mereka pada halaman user. Hasil pengujian dari memperbarui data profil dapat dilihat pada tabel 4.12

Tabel 4. 13 Hasil Pengujian Edit Profil.

Kasus dan Hasil Uji (Data Benar)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Nama: Julian Erza	Berhasil memperbarui data profil.	Sistem berhasil memperbarui data profil.	Diterima
Jenis Kelamin: Laki-laki	Berhasil memperbarui data profil.	Sistem berhasil memperbarui data profil.	Diterima
Tanggal Lahir: 2000-08-08	Berhasil memperbarui data profil.	Sistem berhasil memperbarui data profil.	Diterima
Kasus dan Hasil Uji (Data Kosong)			
Data Masukkan	Harapan	Pengamatan	Kesimpulan
Nama:	Sistem menampilkan	Sistem menampilkan	Diterima

	pesan error	pesan error bahwa field tidak boleh kosong	
Tanggal Lahir:	Sistem menampilkan pesan error	Sistem menampilkan pesan error bahwa field tidak boleh kosong	Diterima

4.2.3. Hasil Pengujian User

Pengujian ini dilakukan dengan cara memberikan kesempatan kepada user untuk mencoba aplikasi secara langsung dan juga memakai smartwatch. Hasil wawancara dari pengujian tersebut dapat dilihat pada tabel berikut.

Tabel 4. 14 Hasil Wawancara dengan Rizky Adha

Pertanyaan	Jawaban
Apakah menurut anda aplikasi ini dapat membantu anda untuk mengetahui kondisi kesehatan jantung anda?	Tentu saja, dikarenakan dengan adanya fitur pengukur denyut jantung, membantu kita mengetahui jumlah denyut jantung
Apakah aplikasi ini membantu anda untuk mengetahui kadar oksigen dalam darah anda.	Setelah dilakukan percobaan, fitur berjalan dengan lancer dan saya dapat mengetahui kadar oksigen
Apakah aplikasi ini dapat membantu anda untuk memprediksi kemungkinan adanya penyakit jantung pada diri anda?	Di Aplikasi ini disediakan fitur untuk mengetahui prediksi penyakit jantung. Sebagai perokok saya bisa melihat prediksi nya berbekal konsumsi rokok

	per hari
Apakah ada kritik dan saran untuk pengembangan aplikasi kedepannya agar lebih baik lagi?	Fitur nya sudah lengkap, namun akan sangat membantu apabila tersedia dalam platform ios

Tabel 4. 15 Hasil Wawancara dengan Dian Ayu

Pertanyaan	Jawaban
Apakah menurut anda aplikasi ini dapat membantu anda untuk mengetahui kondisi kesehatan jantung anda?	Ya aplikasinya bisa mengetahui, kondisi jantung dengan berbagai fitur didalamnya, seperti: fitur pengukur denyut jantung
Apakah aplikasi ini membantu anda untuk mengetahui kadar oksigen dalam darah anda.	Ya saya terbantu dengan informasi saturasi oksigen yang ditampilkan dalam aplikasinya
Apakah aplikasi ini dapat membantu anda untuk memprediksi kemungkinan adanya penyakit jantung pada diri anda?	Ya benar, dengan mengisi data-data yang diperlukan, seperti konsumsi rokok per hari, denyut jantung dan gula darah bisa menampilkan prediksi penyakit jantung
Apakah ada kritik dan saran untuk pengembangan aplikasi kedepannya agar lebih baik lagi?	Kedepannya bisa ditambah fitur pada aplikasinya. Namun design bisa dipertahankan, sebab designya ini mudah digunakan di berbagai kalangan usia, terutama usia lanjut yang susah beradaptasi dengan teknologi semacam ini.

Tabel 4. 16 Hasil Wawancara dengan Anisa Indriani

Pertanyaan	Jawaban
------------	---------

Apakah menurut anda aplikasi ini dapat membantu anda untuk mengetahui kondisi kesehatan jantung anda?	Setelah dicek ini, saya jadi bisa mengatahui denyut jantung saya
Apakah aplikasi ini membantu anda untuk mengetahui kadar oksigen dalam darah anda.	Saya coba gunakan fitur untuk mengetahui kadar oksigen dalam diri saya, ini fitur nya berjalan dengan lancar
Apakah aplikasi ini dapat membantu anda untuk memprediksi kemungkinan adanya penyakit jantung pada diri anda?	Ya bisa memprediksi penyakit jantung. Terlebih dalam aplikasi ini dimudahkan sebab fitur prediksi penyakit jantung ini terintegrasi dengan fitur lainnya. Contohnya, fitur terintegrasi dengan fitur denyut. Sehingga user tidak perlu repot-repot lagi
Apakah ada kritik dan saran untuk pengembangan aplikasi kedepannya agar lebih baik lagi?	Interface aplikasinya menarik, juga memudahkan user salah satunya dengan menampilkan info denyut jantung dan saturasi oksigen di halaman awal sehingga bisa langsung ke highlight. Aplikasi ini sangat berguna dan semoga kedepannya dihadirkan fitur-fitur yang lebih inovatif lainnya.

BAB 5

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan dari hasil implementasi, pengujian, serta wawancara kepada pengguna yang telah dilakukan, maka didapatkan kesimpulan bahwa aplikasi ini fitur didalamnya sudah bisa berjalan sebagaimana mestinya dan sudah dapat membantu user dalam memantau kesehatannya.

5.2. Saran

Untuk menunjang aplikasi ini agar kedepannya dapat membantu user dengan optimal, maka peneliti memberikan saran dengan mempertimbangkan masukan dari peneliti sendiri maupun dari para user. Adapun sarannya adalah sebagai berikut:

1. Selain fitur *primary* dalam aplikasi ini, dapat ditambahkan fitur-fitur tambahan, yang berguna untuk memonitoring kesehatan user.
2. Tampilan pada aplikasi ini bisa dibuat lebih *eye catching*, agar dapat menarik bagi user, terutama dari kalangan anak muda. Tentunya dengan tetap mempertimbangkan dari sisi *user experience*.

Demikian kesimpulan dan saran yang bisa peneliti berikan. Semoga saran-saran tersebut dapat dijadikan masukkan yang bermanfaat dan bisa menjadikan penelitian selanjutnya lebih baik lagi.

DAFTAR PUSTAKA

- [1] M. Rifqi, “Pentingnya Menjaga Kesehatan”, 16 April 2018, [Online], Available:<https://kumparan.com/muhamad-rifki-a/pentingnya-menjaga-kesehatan/1>, [Accessed 30 Januari 2022]
- [2] Satria, “Menjaga Kesehatan Jantung untuk Hidup Lebih Berkualitas”, 7 Oktober 2021, [Online], Available:<https://ugm.ac.id/id/berita/21772-menjaga-kesehatan-jantung-untuk-hidup-lebih-berkualitas>, [Accessed 30 Januari 2022]
- [3] F. R. Makarim, “Beda atau Sama Detak Jantung Normal pada Anak dan Dewasa?”, 3 September 2021, [Online], Available:<https://www.halodoc.com/artikel/beda-atau-sama-detak-jantung-normal-pada-anak-dan-dewasa>, [Accessed 31 Januari 2022]
- [4] K. Adrian, “Mengetahui Nilai Saturasi Oksigen dan Cara Meningatkannya”, 9 Juli 2021, [Online], Available:<https://www.alodokter.com/mengetahui-nilai-saturasi-oksigen-dan-cara-meningatkannya>, [Accessed 31 Januari 2022]
- [6] A. S. Utomo, E. H. P. Negoro, M. Sofie, “Monitoring Heart Rate dan Saturasi Oksigen Melalui Smartphone”, Jurnal SIMETRIS, Vol. 10, No. 1, 2019.
- [7] M. Nareza, “Penting Diketahui, Ini Kadar Oksigen Normal dalam Darah”, 12 Februari 2021, [Online], Available: <https://www.alodokter.com/penting-diketahui-ini-kadar-oksigen-normal-dalam-darah>, [Accessed 7 Februari 2022]
- [9] S. Pers, “Ini Pentingnya Memantau Kesehatan saat Pandemi”, 28 Juni 2021, [Online], Available: <https://koranbernas.id/ini-pentingnya-memantau-kesehatan-saat-pandemi>, [Accessed 30 Januari 2022].
- [10] I. F. Faisal, A. P. Kharisma, Sutrisno, “Pengembangan Aplikasi Pendekripsi Kantuk Pada Pengendara Kendaraan Bermotor Dengan Menggunakan Sensor Detak Jantung Pada Smartwatch”, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, Vol. 3, No.10, pp. 9568-9578, 2019.

- [11] V. Yonanto, D. W. G. Wisana, T. Rahmawati, “Pemantauan SpO₂ Melalui Aplikasi Android di Mobile Phone”, TEKNOGES, Vol. 12, No. 2, pp. 21-27, 2019.
- [12] D. P. Nugroho, R. Munadi, I. H. Santoso, “Sistem Pemantauan Kondisi Detak Jantung Berbasis Internet of Things Menggunakan Sensor EKG Dengan Media Aplikasi Android”, e-Proceeding of Engineering, Vol. 8, No. 5, pp. 5530-5536, 2021.
- [13] A. Andriani, R. Hartono, “Saturasi Oksigen Dengan Pulse Oximetry Dalam 24 Jam Pada Pasien Dewasa Terpasang Ventilator Di Ruang ICU Rumah Sakit Panti Wilasa Citarum Semarang”, Jendela Nursing Journal, Vol. 2, No. 1, 2013.
- [14] K. Adrian, “Pentingnya Oximeter bagi Pasien Isolasi Mandiri COVID-19”, 1 Februari 2021, [Online], Available: <https://www.alodokter.com/pentingnya-oximeter-bagi-pasien-isolasi-mandiri-covid-19>, [Accessed 28 Januari 2022].
- [15] A. A. Wahid, “Analisis Metode Waterfall Untuk Pengembangan Sistem Informasi”, Jurnal Ilmu-ilmu Informatika dan Manajemen STMIK, 2020.
- [16] Nurhidayati, A. M. Nur, “Pemanfaatan Aplikasi Android Dalam Rancang Bangun Sistem Informasi Persebaran Indekos di Wilayah Pancor Kabupaten Lombok Timur”, Infotek: Jurnal Informatika dan Teknologi, Vol. 4, No. 1, 2021.
- [17] M. MacGill, “Normal Resting Heart Rate”, 19 Januari 2021, [Online], Available: <https://www.medicalnewstoday.com/articles/235710#normal-resting-heart-rate>, [Accessed 6 Juli 2022]

DAFTAR LAMPIRAN

Lampiran A Hasil Wawancara

Untuk membuktikan bahwa pembangunan sistem telah berhasil dilakukan adalah dengan cara melakukan wawancara kepada para pengguna sistem yang dipilih sebagai pengujian sistem.

Lampiran Hasil Wawancara dengan Narasumber Pertama

Wawancara pertama dilakukan dengan narasumber pertama. Untuk data diri narasumber pertama, dapat dilihat pada tabel A.1 berikut.

Tabel A. 1 Data Diri Narasumber Pertama

Data Diri Narasumber	
Nama Lengkap	Rizky Adha
Alamat Email	rizqyadha13@gmail.com
Pekerjaan	Wordpress Developer
Nomor Kontak	081953803436

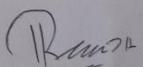
Untuk membuktikan bahwa memang wawancara benar dilakukan, peneliti juga melampirkan foto ketika melakukan wawancara, dengan foto form hasil pengujian yang diisi langsung oleh narasumber dan juga ditanda tangani oleh narasumber. Berikut adalah bukti-bukti tersebut.

FORM HASIL PENGUJIAN APLIKASI

Nama Lengkap	Rizqi Adha	
Alamat Email	rizaadha13@gmail.com	
Pekerjaan	Wordpress Developer	
Nomor Kontak	0819 5380 3936	
Pernyataan	Setuju	Tidak Setuju
Aplikasi ini membantu saya untuk memonitoring kesehatan khususnya untuk denyut jantung dan saturasi oksigen.	✓	
Aplikasi ini memberikan informasi terkait penanganan jika kondisi denyut jantung atau saturasi oksigen saya berada diluar normal.	✓	
Pertanyaan	Jawaban	
Apakah ada kritik dan saran untuk pengembangan aplikasi agar kedepannya bisa lebih baik lagi?	Fitur yang dapat membantu kesehatan ini, bisa diperbaiki lagi, seperti tersedian di platform iOS	

Dengan ini, saya menyatakan bahwa jawaban yang saya isi adalah benar jawaban berdasarkan pengalaman saya dalam menggunakan aplikasi ini tanpa ada paksaan dari pihak manapun.

Bandung, 17 Agustus 2022



Narasumber

Gambar A. 1 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan



Gambar A. 2 Narasumber sedang mengisi form bukti pengujian

Lampiran Hasil Wawancara dengan Narasumber Kedua

Wawancara pertama dilakukan dengan narasumber kedua. Untuk data diri narasumber pertama, dapat dilihat pada tabel A.2 berikut.

Tabel A. 2 Data Diri Narasumber Kedua

Data Diri Narasumber	
Nama Lengkap	Dian Ayu Suci Maharani
Alamat Email	dianayusuci12@gmail.com
Pekerjaan	Software Quality Assurance
Nomor Kontak	081221894983

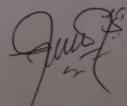
Untuk membuktikan bahwa memang wawancara benar dilakukan, peneliti juga melampirkan foto ketika melakukan wawancara, dengan foto form hasil pengujian yang diisi langsung oleh narasumber dan juga ditanda tangani oleh narasumber. Berikut adalah bukti-bukti tersebut.

FORM HASIL PENGUJIAN APLIKASI

Nama Lengkap	: Dian Ayu Suci Maharani	
Alamat Email	: dianayusuci12@gmail.com	
Pekerjaan	: SQA	
Nomor Kontak	: 081221894983	
Pernyataan	Setuju	Tidak Setuju
Aplikasi ini membantu saya untuk memonitoring kesehatan khususnya untuk denyut jantung dan saturasi oksigen.	✓	
Aplikasi ini memberikan informasi terkait penanganan jika kondisi denyut jantung atau saturasi oksigen saya berada diluar normal.	✓	
Pertanyaan	Jawaban	
Apakah ada kritik dan saran untuk pengembangan aplikasi agar kedepannya bisa lebih baik lagi?	Aplikasinya sudah bagus semoga bisa berkembang untuk alat deteksi lainnya . banyak , semua user bisa lebih mencari aplikasi lain untuk menunjang pengukuran lainnya . Hanya cukup di 1 aplikasi ini saja	

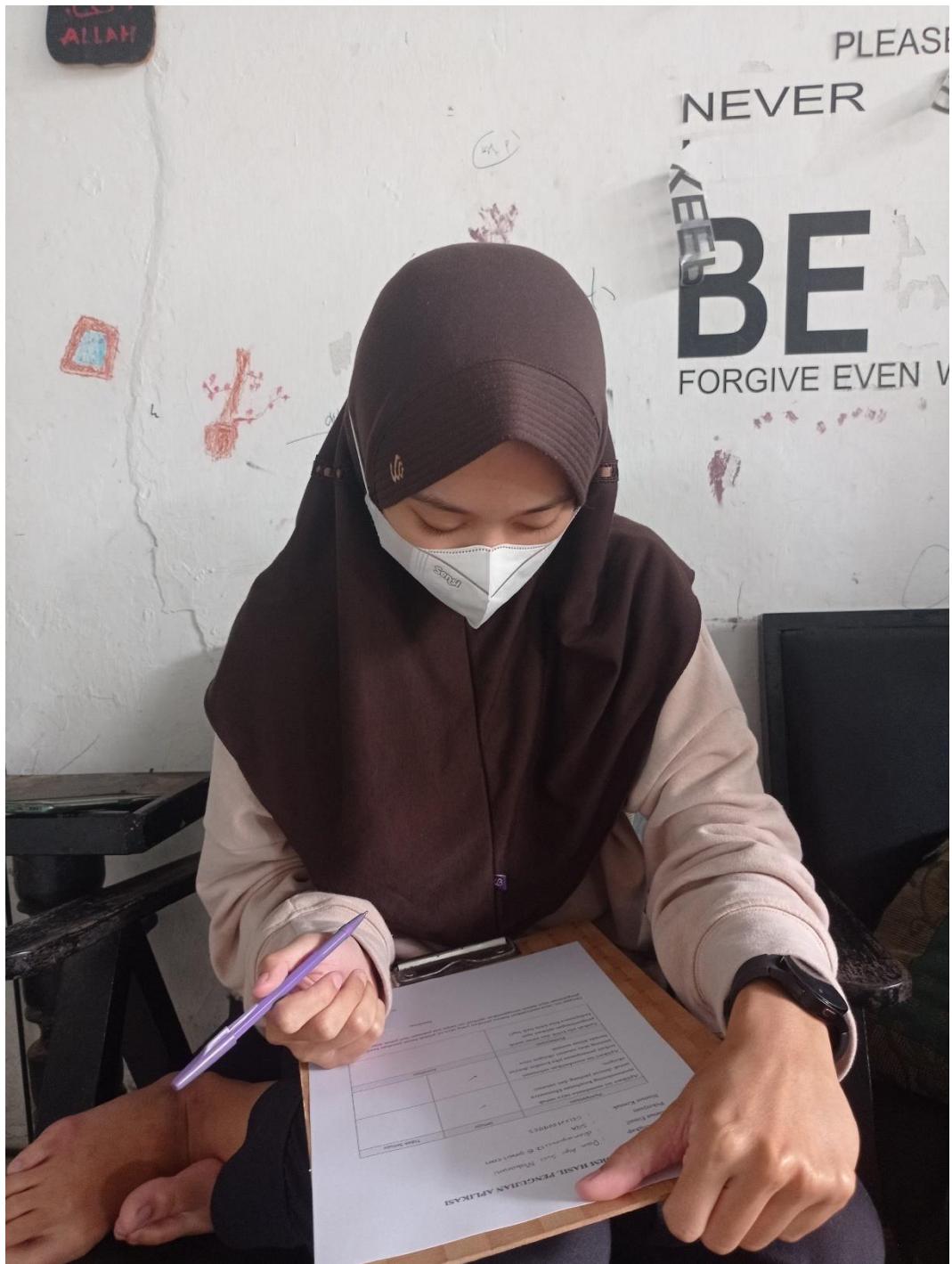
Dengan ini, saya menyatakan bahwa jawaban yang saya isi adalah benar jawaban berdasarkan pengalaman saya dalam menggunakan aplikasi ini tanpa ada paksaan dari pihak manapun.

Bandung, 20 Agustus 2022



Narasumber

Gambar A. 3 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan



Gambar A. 4 Narasumber sedang mengisi form bukti pengujian

Lampiran Hasil Wawancara dengan Narasumber Ketiga

Wawancara pertama dilakukan dengan narasumber ketiga. Untuk data diri narasumber pertama, dapat dilihat pada tabel A.3 berikut.

Tabel A. 3 Data Diri Narasumber Ketiga

Data Diri Narasumber	
Nama Lengkap	Anisa Indriani
Alamat Email	anisaindriani15@gmail.com
Pekerjaan	Software Quality Assurance
Nomor Kontak	085159160706

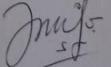
Untuk membuktikan bahwa memang wawancara benar dilakukan, peneliti juga melampirkan foto ketika melakukan wawancara, dengan foto form hasil pengujian yang diisi langsung oleh narasumber dan juga ditanda tangani oleh narasumber. Berikut adalah bukti-bukti tersebut.

FORM HASIL PENGUJIAN APLIKASI

Nama Lengkap	: Anisa Indriani	
Alamat Email	: anisaindriani15@gmail.com	
Pekerjaan	: Software qual	
Nomor Kontak	: 08515916 0706	
Pernyataan	Setuju	Tidak Setuju
Aplikasi ini membantu saya untuk memonitoring kesehatan khususnya untuk denyut jantung dan saturasi oksigen.	✓	
Aplikasi ini memberikan informasi terkait penanganan jika kondisi denyut jantung atau saturasi oksigen saya berada diluar normal.	✓	
Pertanyaan	Jawaban	
Apakah ada kritik dan saran untuk pengembangan aplikasi agar kedepannya bisa lebih baik lagi?	Aplikasi ini sangat berguna dan semoga kedepannya dihadirkan fitur-fitur yang lebih inovatif lainnya.	

Dengan ini, saya menyatakan bahwa jawaban yang saya isi adalah benar jawaban berdasarkan pengalaman saya dalam menggunakan aplikasi ini tanpa ada paksaan dari pihak manapun.

Bandung, 19 Agustus 2022



Narasumber

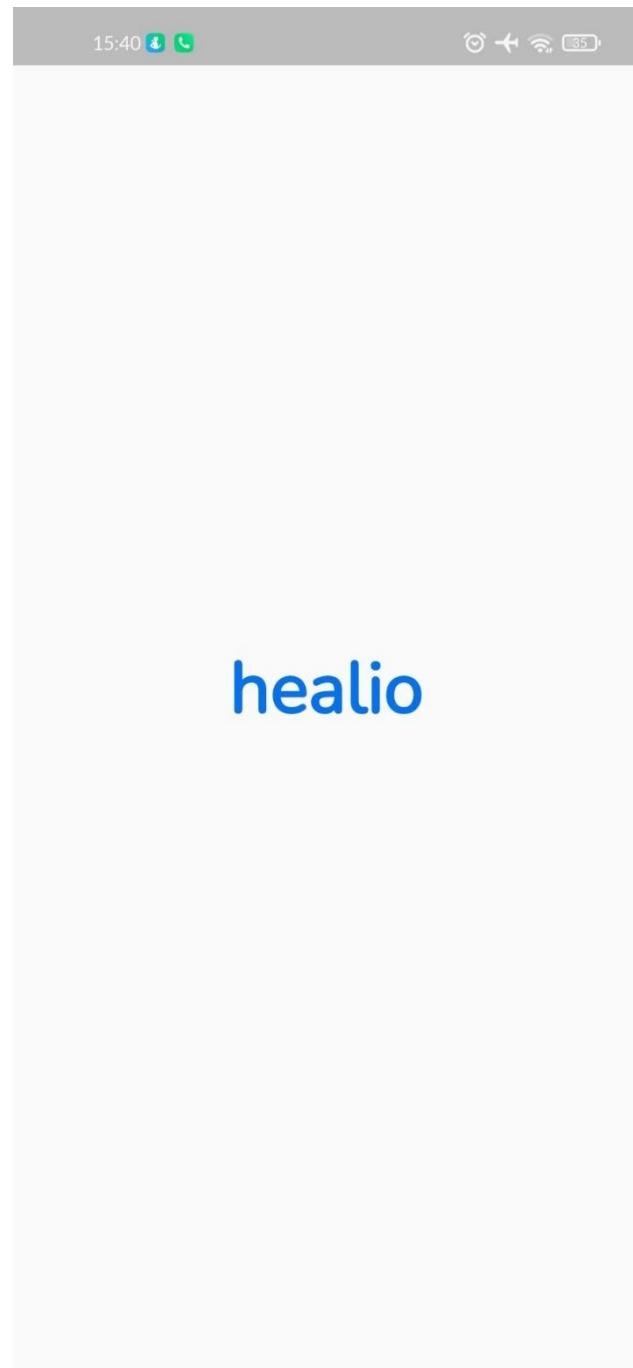
Gambar A. 5 Form hasil pengujian yang membuktikan bahwa pengujian dan wawancara benar telah dilakukan



Gambar A. 6 Narasumber sedang mengisi form bukti pengujian

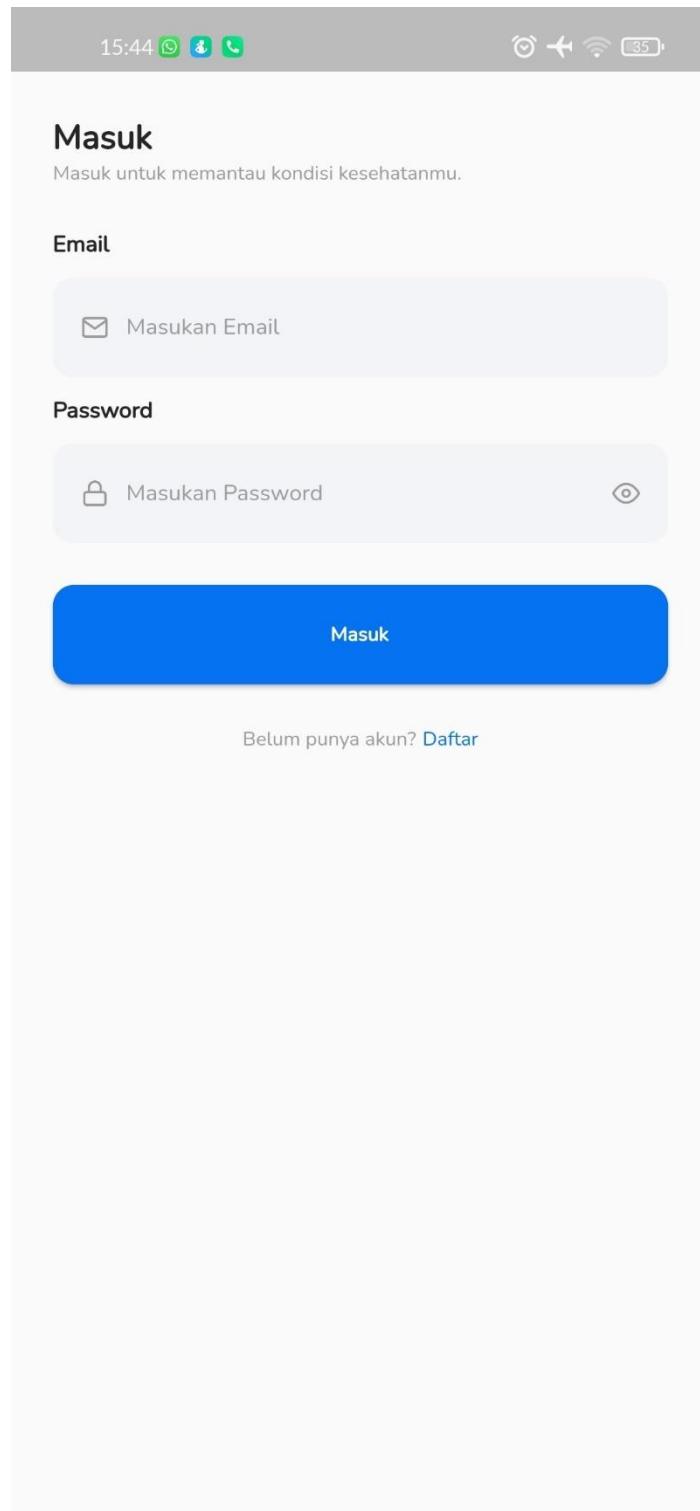
Lampiran B Screenshot Aplikasi

1. *Screenshot Halaman Splash Page*



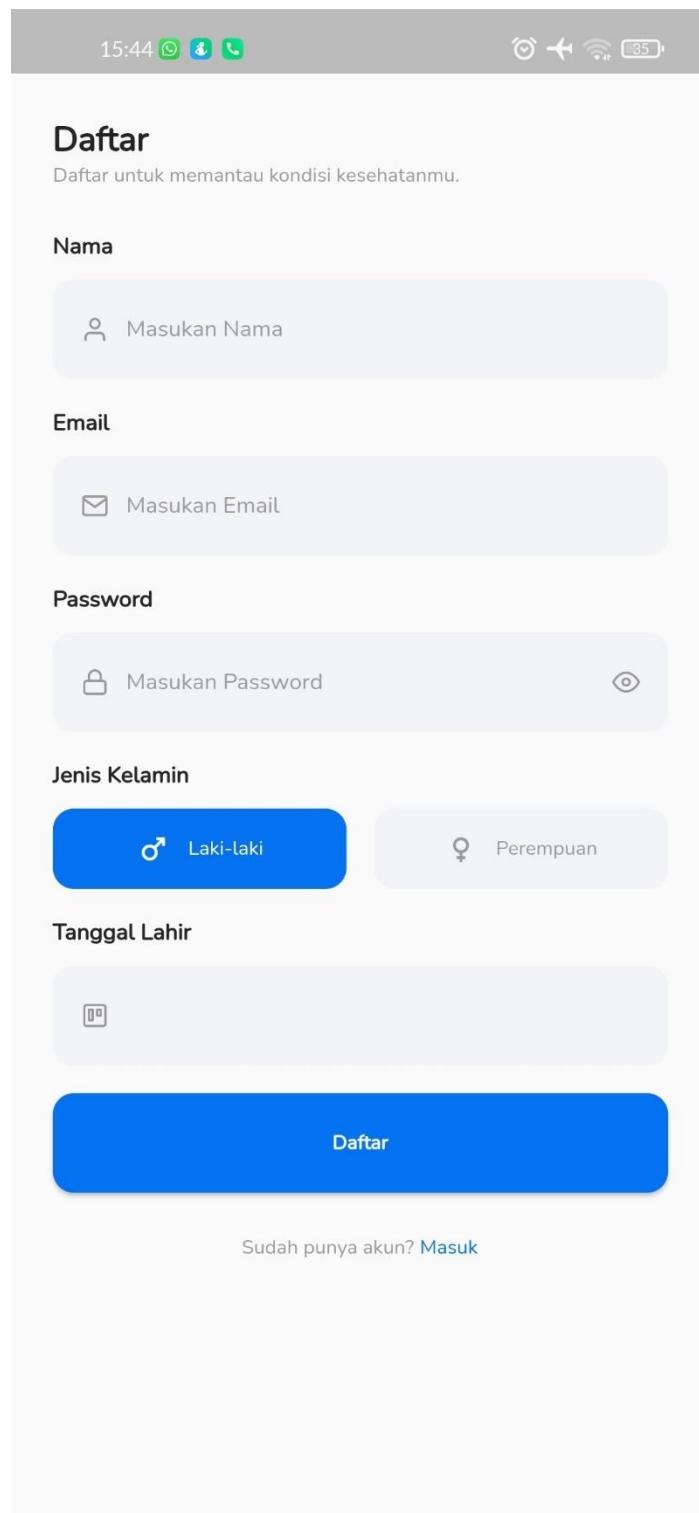
Gambar B. 1 Screenshot Halaman Splash Page

2. *Screenshot Halaman Login*



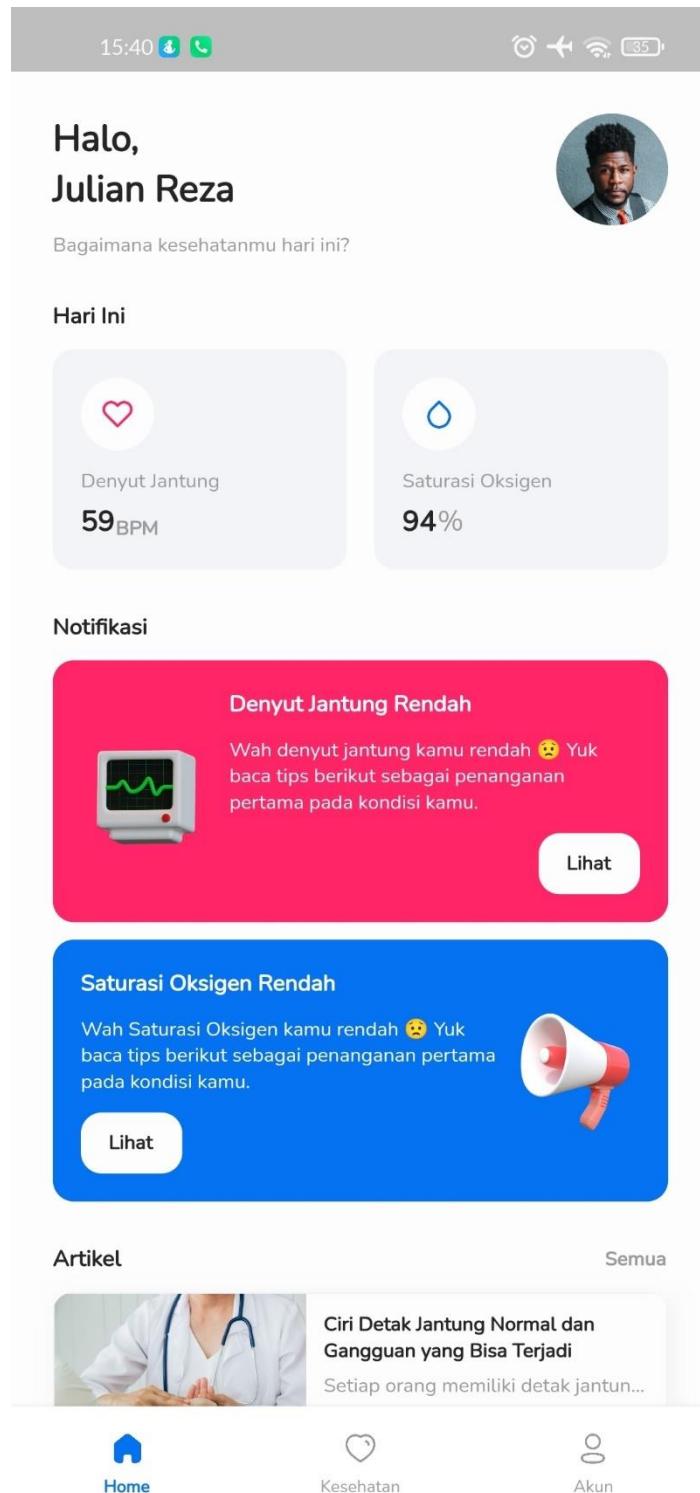
Gambar B. 2 Screenshot Halaman Login

3. *Screenshot Halaman Register*



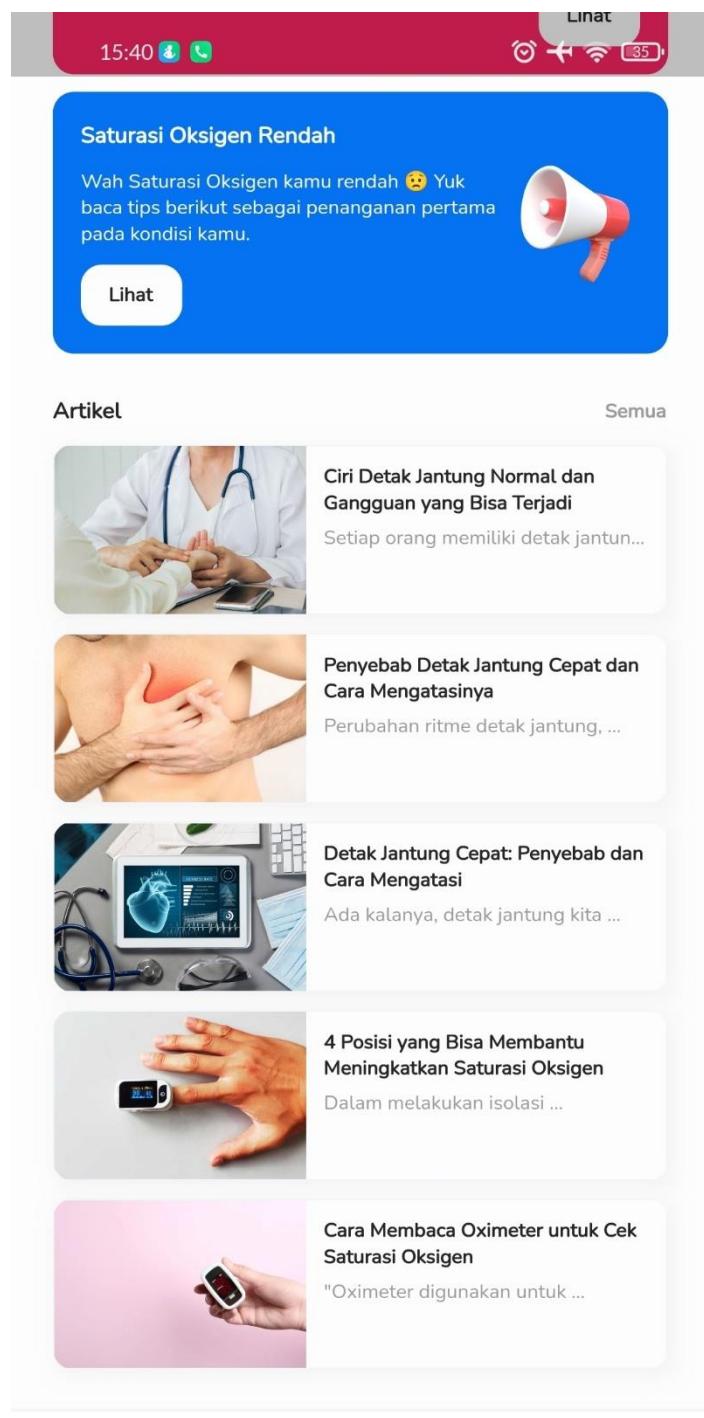
Gambar B. 3 Screenshot Halaman Register

4. Screenshot Halaman Home



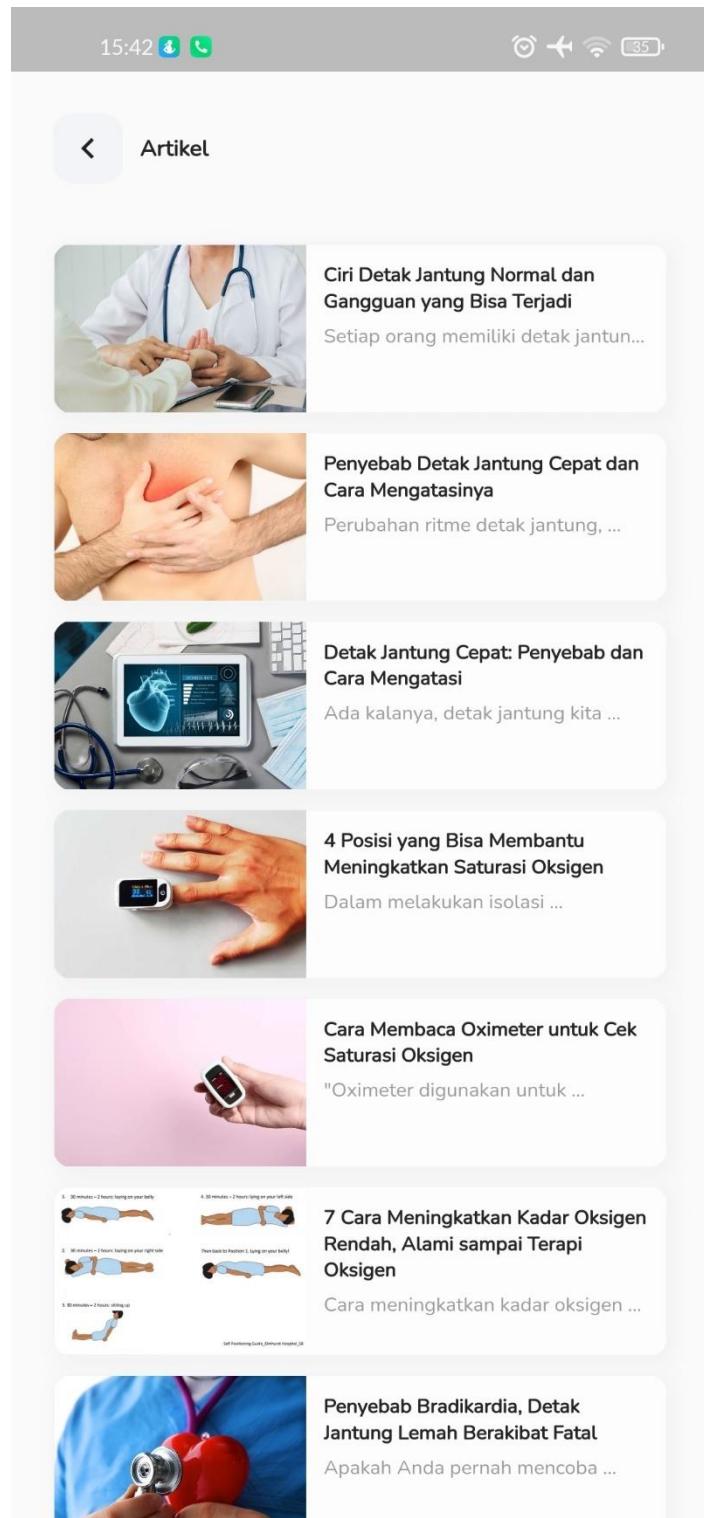
Gambar B. 4 Screenshot Halaman Home

5. Screenshot Halaman Home 2



Gambar B. 5 Screenshot Halaman Home 2

6. Screenshot Halaman Artikel



Gambar B. 6 Screenshot Halaman Artikel

7. Screenshot Halaman Artikel Webview

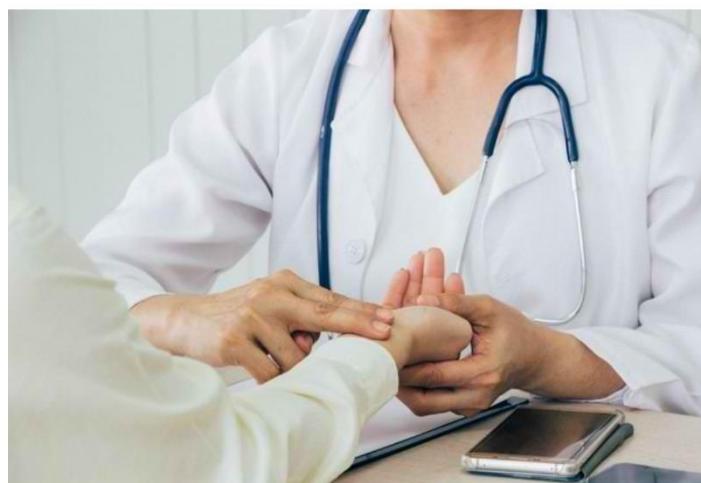


Kesehatan

Ciri Detak Jantung Normal dan Gangguan yang Bisa Terjadi

Setiap orang memiliki detak jantung normal yang berbeda-beda dan dapat dipengaruhi oleh berbagai hal, termasuk kondisi kesehatan tertentu. Nah, dengan mengetahui detak jantung normal, Anda pun bisa lebih waspada terhadap kemungkinan adanya gangguan pada jantung.

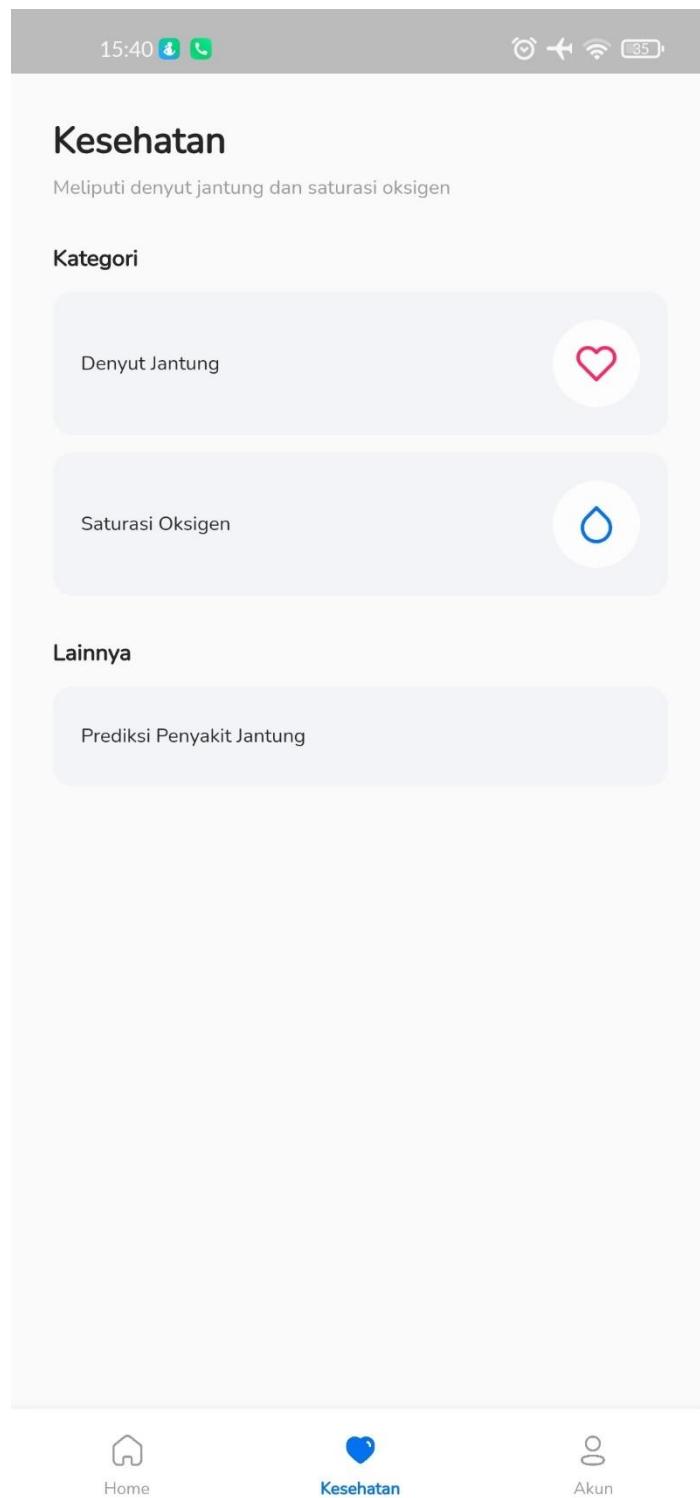
Detak jantung diatur oleh sistem listrik pada organ jantung. Detak jantung yang normal akan terdengar seirama dan sama setiap ketukannya. Hal ini menandakan jantung berfungsi dengan baik.



Sementara itu, [detak jantung abnormal](#) akan terdengar tidak beraturan dan bahkan terdengar suara bising di luar suara detak jantung utama.

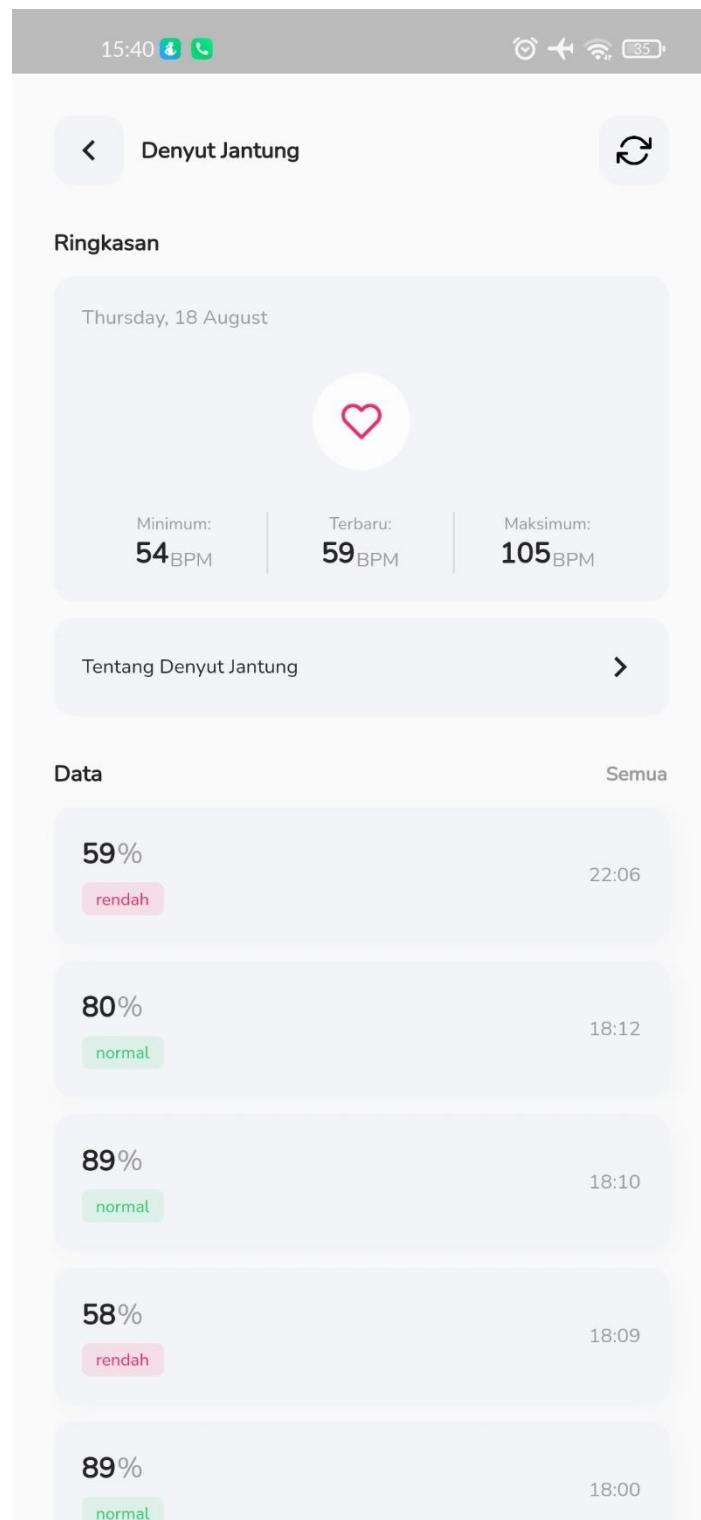
Gambar B. 7 Screenshot Halaman Artikel Webview

8. *Screenshot* Halaman Kesehatan



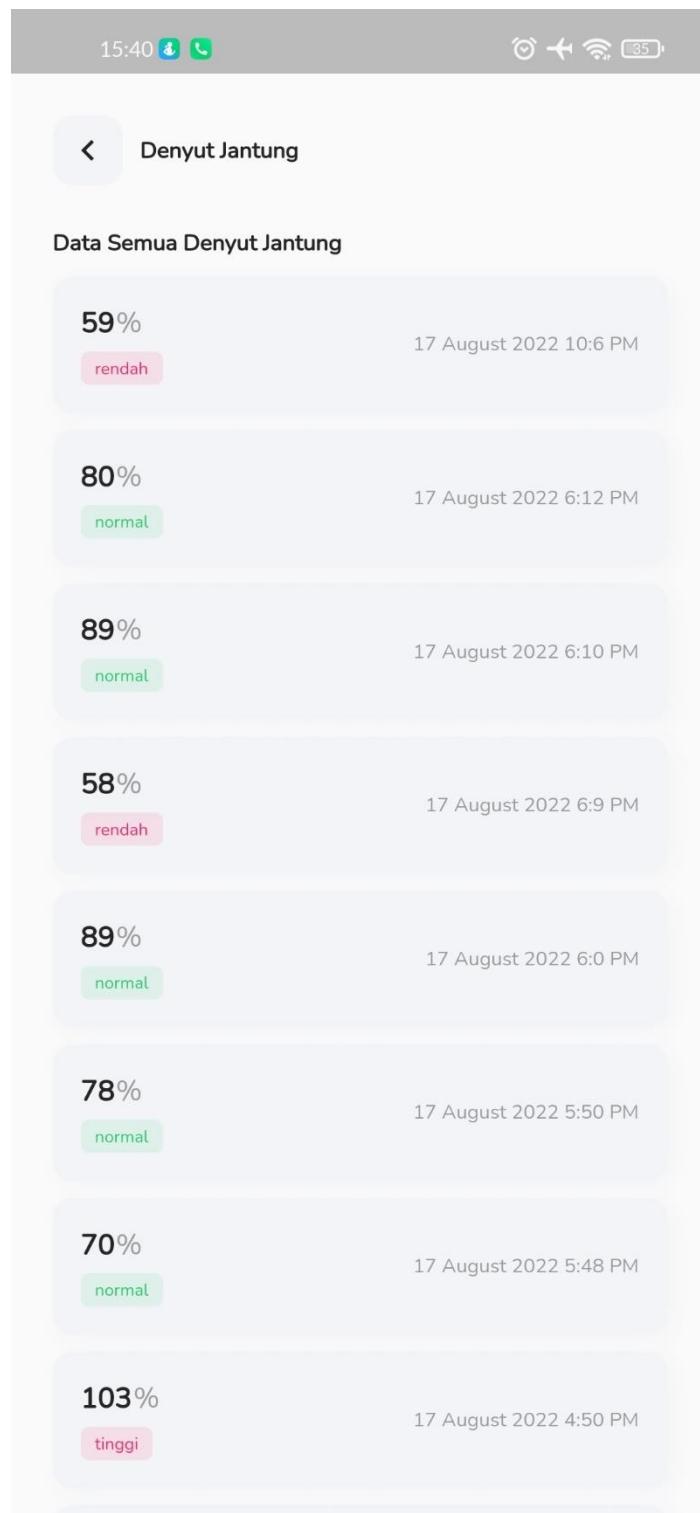
Gambar B. 8 Screenshot Halaman Kesehatan

9. Screenshot Halaman Denyut Jantung



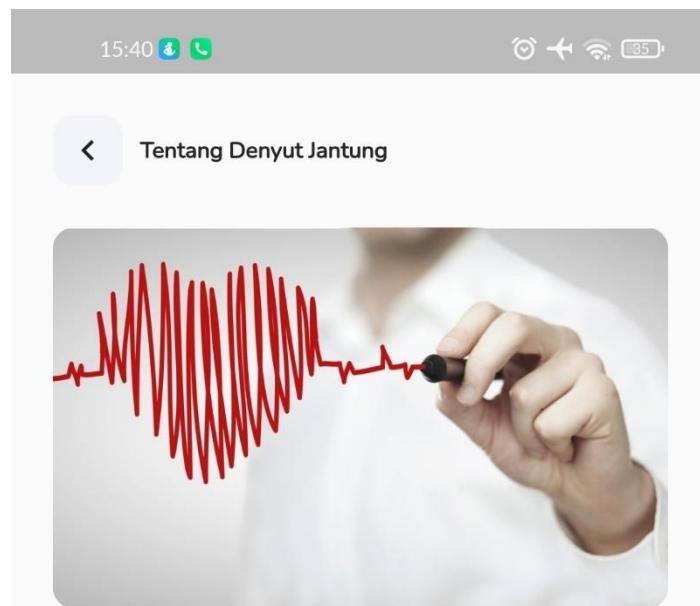
Gambar B. 9 Screenshot Halaman Denyut Jantung

10. *Screenshot Halaman Data Denyut Jantung*



Gambar B. 10 Screenshot Halaman Data Denyut Jantung

11. Screenshot Halaman Tentang Denyut Jantung



The screenshot shows a mobile application interface. At the top, there is a grey header bar with the time '15:40' and several icons. Below the header, the title 'Tentang Denyut Jantung' is displayed in bold black text next to a back arrow icon. The main content area features a large, stylized red ECG waveform on a white background. A person's hand, wearing a white shirt, is shown holding a black marker and drawing the waveform. Below the image, the section title 'Detak Jantung' is written in bold black text. Underneath the title, there is a paragraph of text explaining the function of the heart. Further down, another paragraph discusses factors affecting heart rate. At the bottom of the content area, there is a caption 'Tabel Detak Jantung Berdasarkan Usia' followed by a table.

Detak Jantung

Jantung merupakan organ vital pada tubuh manusia. Fungsi jantung adalah memompa darah ke seluruh tubuh, sehingga berbagai organ dan sistem tubuh Anda bisa bekerja sebagaimana mestinya.

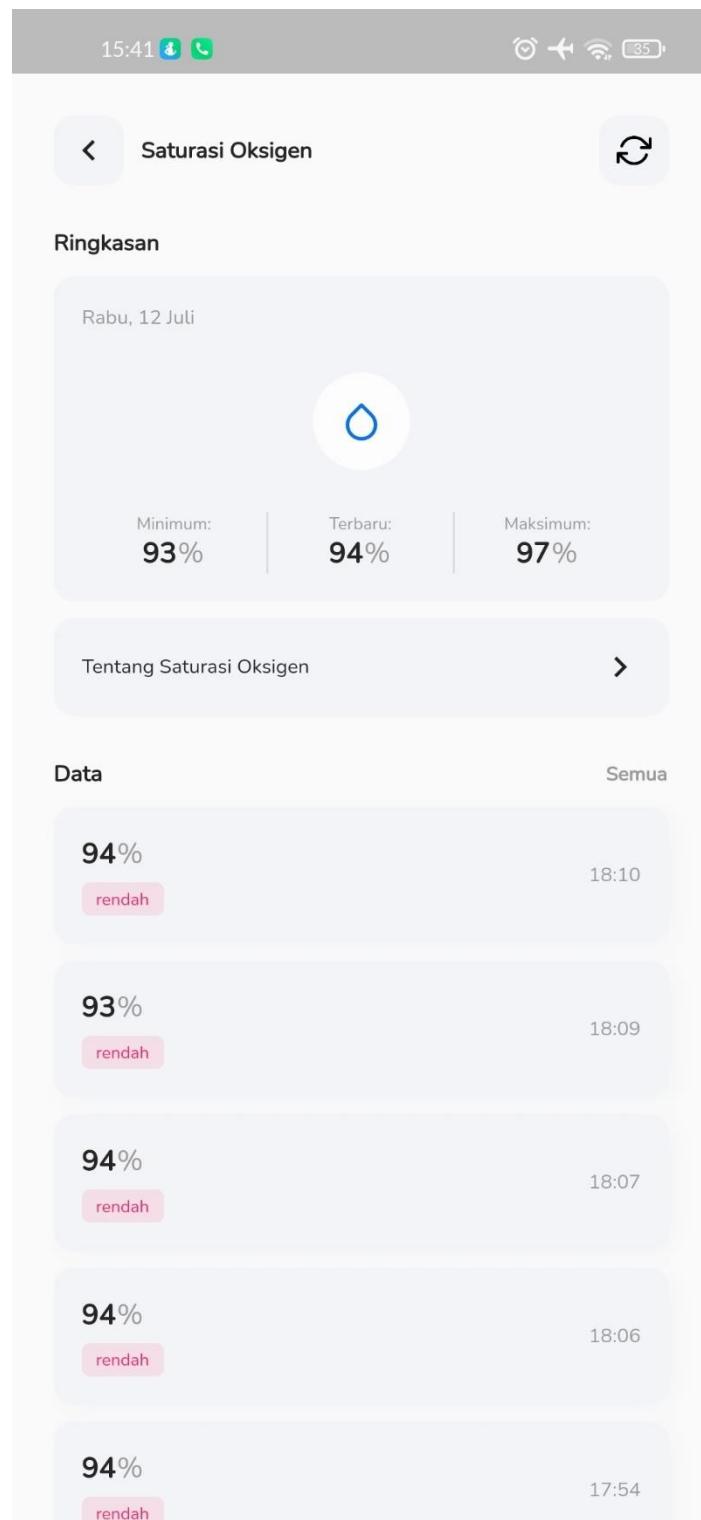
Selain tekanan darah, salah satu indikator penting dari kesehatan jantung adalah detak jantung. Detak jantung adalah berapa kali jantung Anda berdenyut dalam satu menit. Adapun detak jantung seseorang dipengaruhi oleh berbagai faktor, seperti usia, ukuran tubuh, kondisi jantung, cuaca atau temperatur udara, aktivitas fisik, emosi, dan obat-obatan tertentu.

Tabel Detak Jantung Berdasarkan Usia

Usia	Normal (bpm)
Bayi	70 - 130
Anak - anak	80 - 110
Dewasa	60 - 100

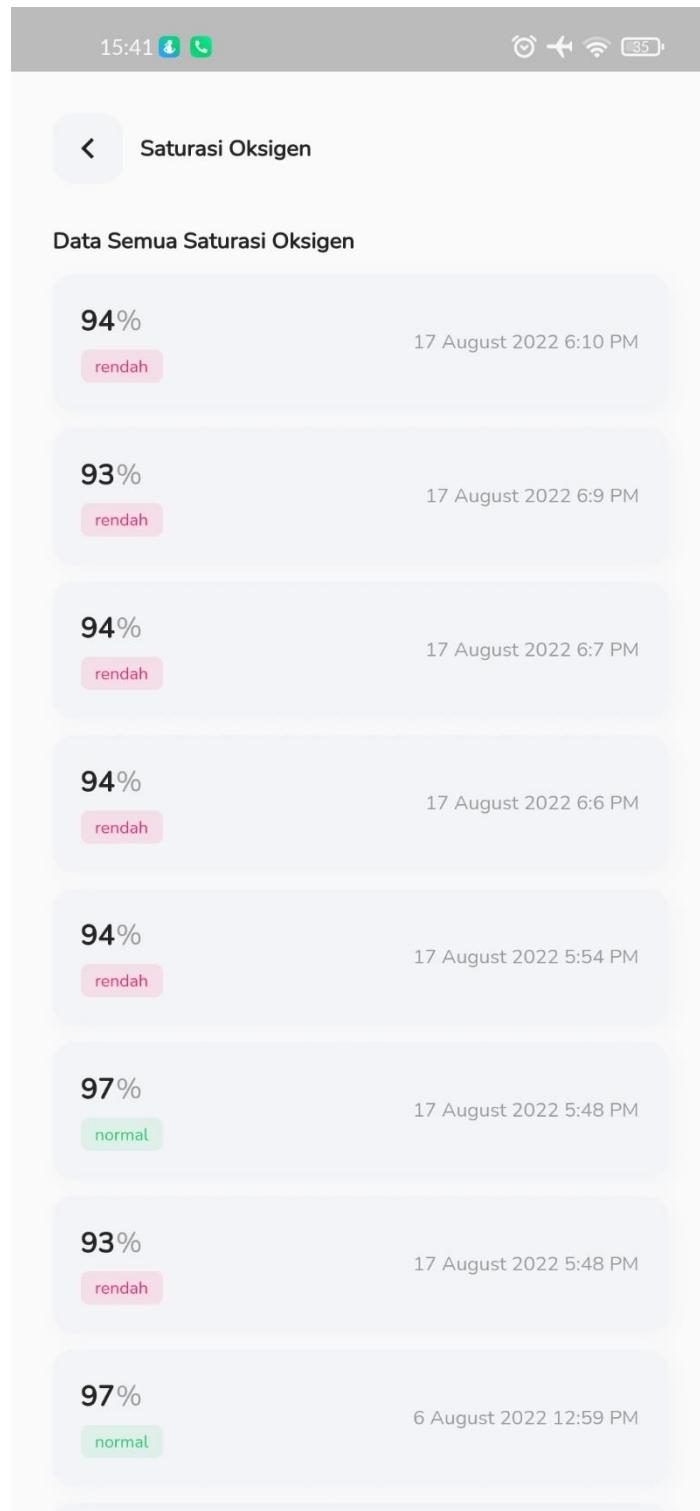
Gambar B. 11 Screenshot Halaman Tentang Denyut Jantung

12. Screenshot Halaman Saturasi Oksigen



Gambar B. 12 Screenshot Halaman Saturasi Oksigen

13. *Screenshot Halaman Data Saturasi Oksigen*



Gambar B. 13 Screenshot Halaman Data Saturasi Oksigen

14. Screenshot Halaman Tentang Saturasi Oksigen

The screenshot shows a mobile phone screen with a white background. At the top, there is a navigation bar with icons for time (15:41), signal strength, battery level (35%), and other status indicators. Below the navigation bar, the title "Tentang Saturasi Oksigen" is displayed in bold black text, preceded by a back arrow icon. The main content area features a photograph of a person's hand holding a blue and white pulse oximeter on their index finger. The device's screen displays numerical values. Below the photo, the heading "Saturasi Oksigen" is centered in bold black text. The following text provides information about oxygen saturation:

Saturasi oksigen merupakan nilai yang menunjukkan kadar oksigen di dalam darah. Nilai ini sangat berpengaruh terhadap berbagai fungsi organ dan jaringan tubuh. Pengukuran nilai saturasi oksigen dapat dilakukan dengan 2 cara, yakni dengan analisis gas darah (AGD) atau menggunakan alat oximeter.

Hasil pengukuran saturasi oksigen yang dilakukan dengan analisis gas darah ditunjukkan dengan istilah PaO₂ (tekanan parsial oksigen). Sementara itu, hasil pengukuran saturasi oksigen dengan menggunakan oximeter ditunjukkan dengan istilah SpO₂.

Saturasi Oksigen Normal

Berikut adalah nilai saturasi oksigen normal pada orang dengan kondisi paru-paru yang sehat atau tidak memiliki kondisi medis tertentu:

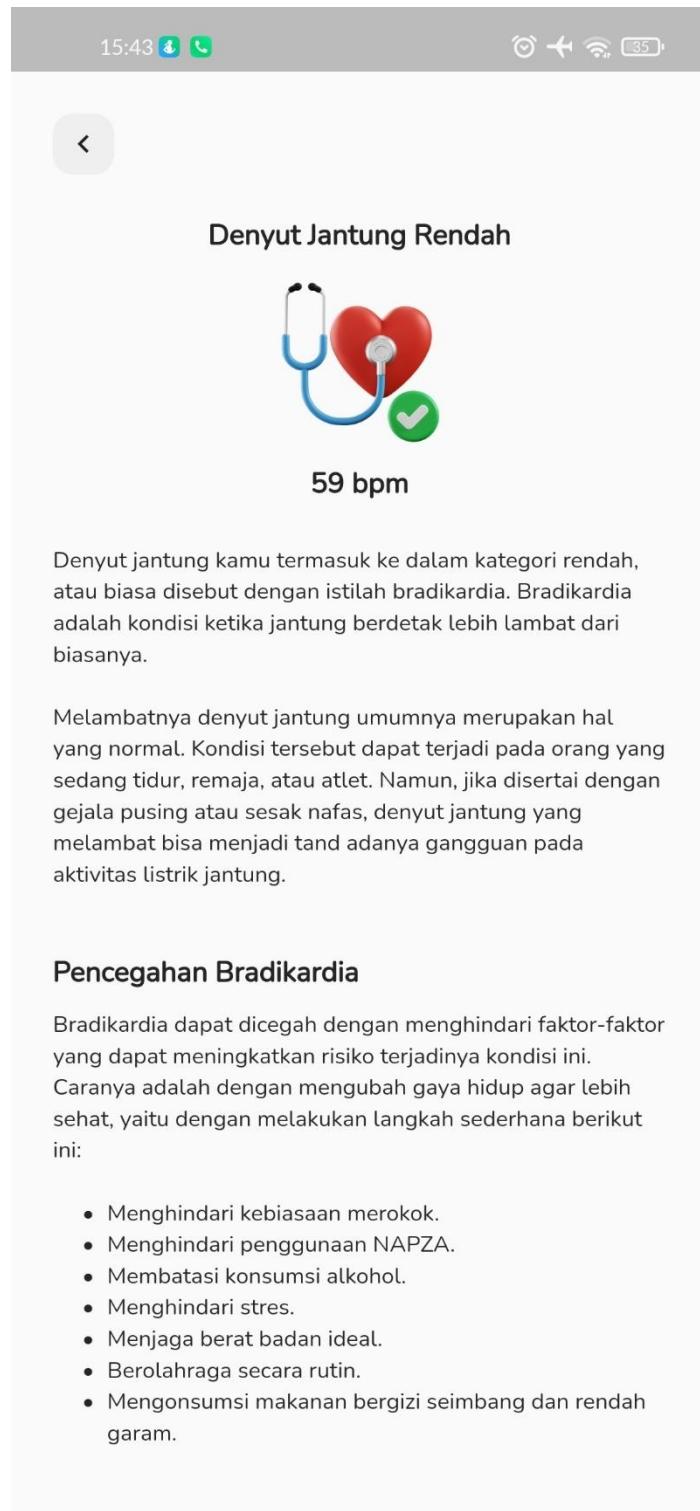
- Analisis gas darah (PaO₂): 80–100 mmHg
- Oximeter (SpO₂): 95–100%

Saturasi Oksigen Rendah

Berikut adalah kriteria nilai saturasi oksigen rendah atau di

Gambar B. 14 Screenshot Halaman Tentang Saturasi Oksigen

15. Screenshot Halaman Penanganan Denyut Jantung Rendah



Gambar B. 15 Screenshot Halaman Penanganan Denyut Jantung Rendah

16. Screenshot Halaman Penanganan Denyut Jantung Tinggi

The screenshot shows a mobile application interface. At the top, there is a header bar with the time (16:25), battery level (39%), and signal strength icons. Below the header, the title "Denyut Jantung Tinggi" is displayed. Underneath the title is a graphic of a red heart with a blue stethoscope wrapped around it, and a green checkmark icon. The text "105 bpm" is shown below the graphic. The main content area contains two paragraphs of text about atrial fibrillation and its complications. Below this, a section titled "Penanganan Denyut Jantung Tinggi" is present, followed by two more paragraphs of text about managing atrial fibrillation. At the bottom, there is a section titled "Tindakan" with a single paragraph of text.

Denyut jantung kamu termasuk ke dalam kategori tinggi, atau biasa disebut dengan istilah takikardia. Takikardia terjadi jika denyut jantung melebihi 100 kali per menit atau bpm. Kondisi ini terjadi karena adanya gangguan listrik di jantung yang berperan dalam mengontrol irama denyut jantung.

Takikardia bisa muncul tanpa menimbulkan komplikasi. Namun, jika tidak ditangani, denyut jantung yang tinggi bisa menyebabkan komplikasi serius, seperti stroke, gagal jantung, henti jantung, dan bahkan kematian.

Penanganan Denyut Jantung Tinggi

Denyut jantung tinggi yang terjadi bukan karena penyakit, umumnya tidak membutuhkan pengobatan karena dapat membaik dengan sendirinya. Jika disebabkan oleh kondisi medis tertentu, penanganan denyut jantung tinggi akan disesuaikan dengan faktor penyebabnya.

Penanganan yang dilakukan bertujuan untuk memperlambat denyut jantung tinggi hingga kembali dalam batas normal, mencegah denyut jantung tinggi kembali, menekan risiko komplikasi, dan mengobati penyakit mendasar yang dapat menyebabkan takikardia.

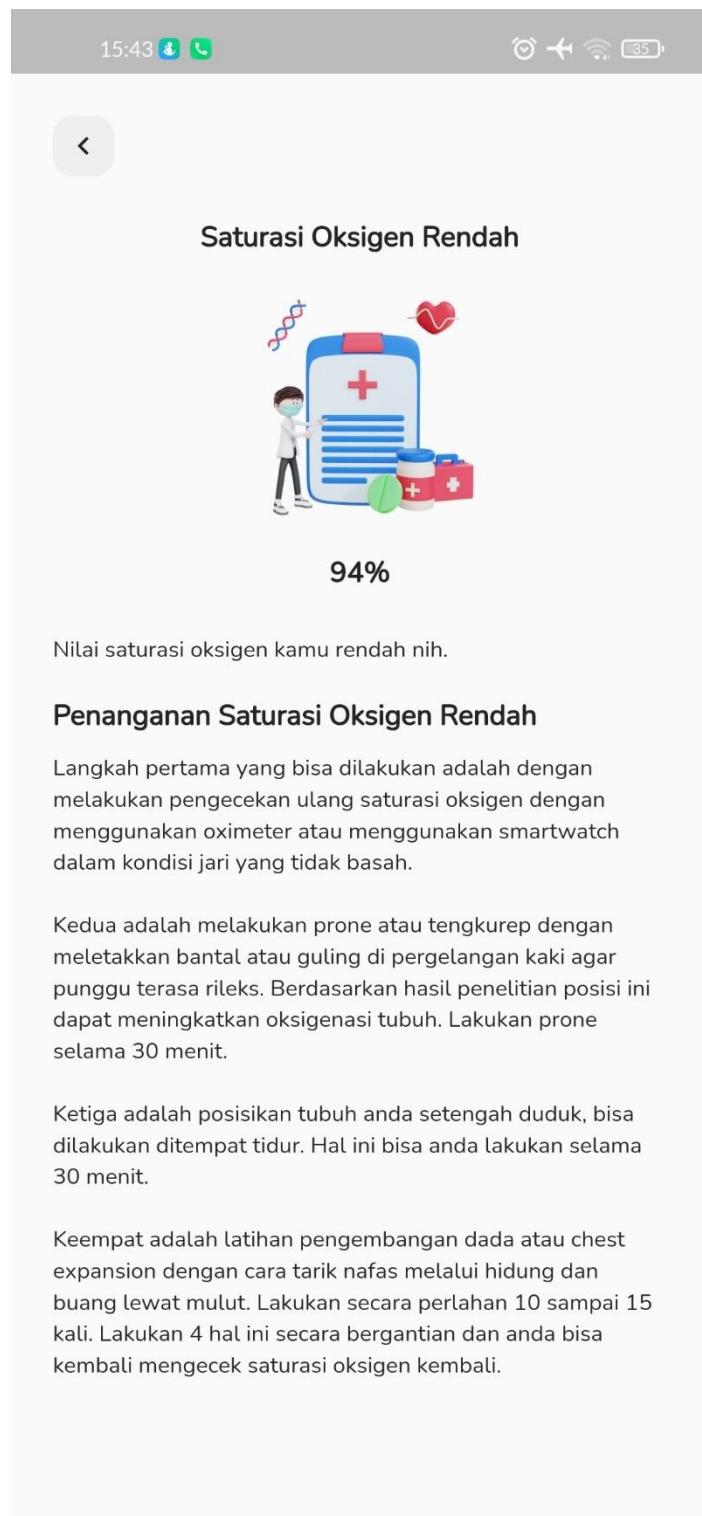
Penanganan untuk takikardia meliputi:

Tindakan

Untuk meredakan denyut jantung tinggi, kamu bisa melakukan tindakan Manuver Vagal, yaitu melakukan gerakan seperti batuk, mengejan sebagaimana tengah

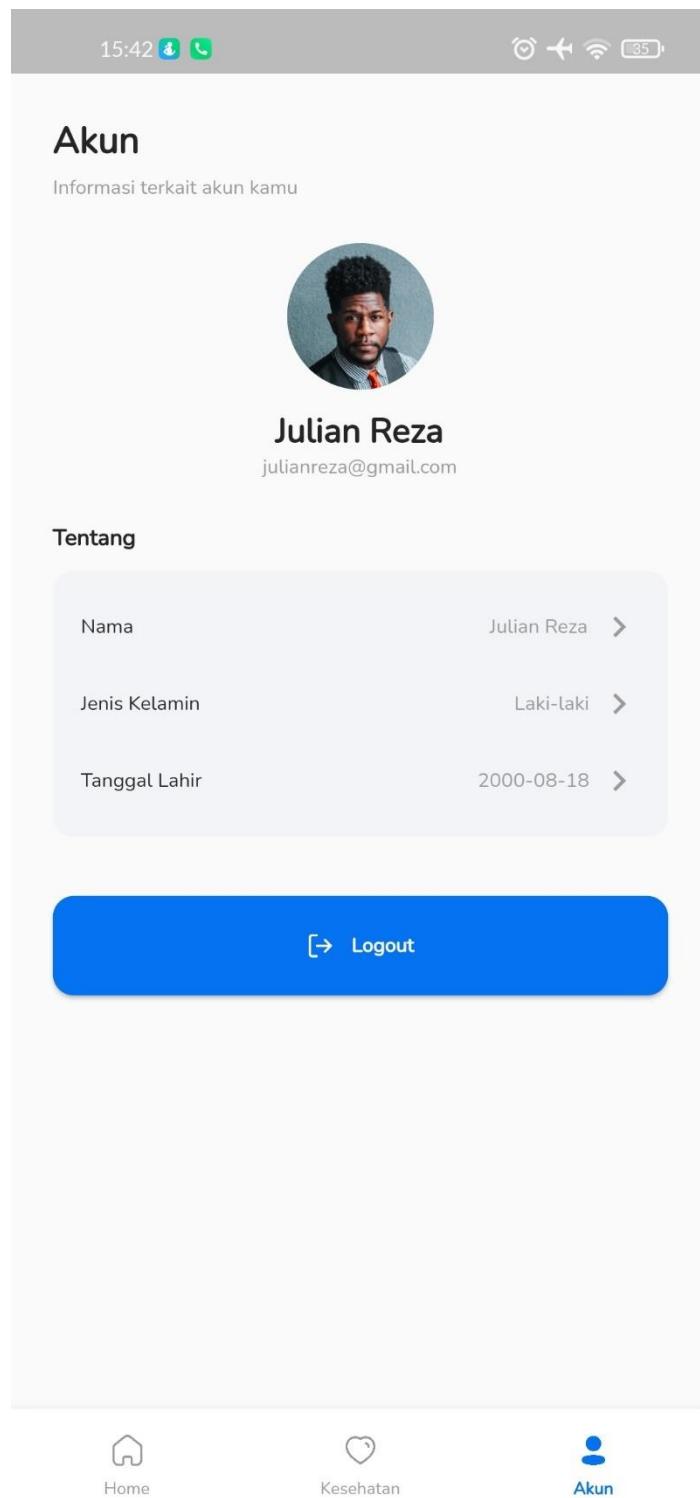
Gambar B. 16 Screenshot Halaman Penanganan Denyut Jantung Tinggi

17. Screenshot Halaman Penanganan Saturasi Oksigen Rendah



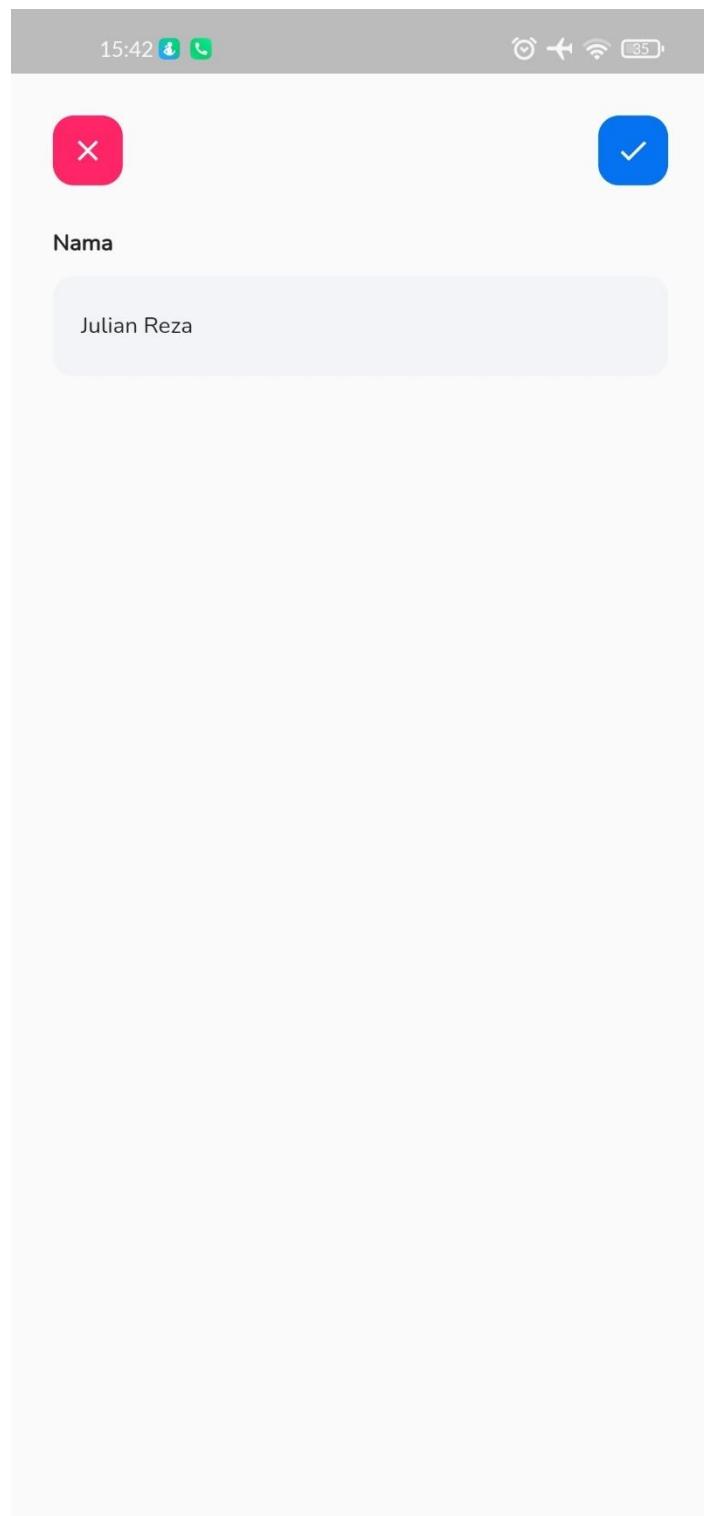
Gambar B. 17 Screenshot Halaman Penanganan Saturasi Oksigen Rendah

18. Screenshot Halaman User



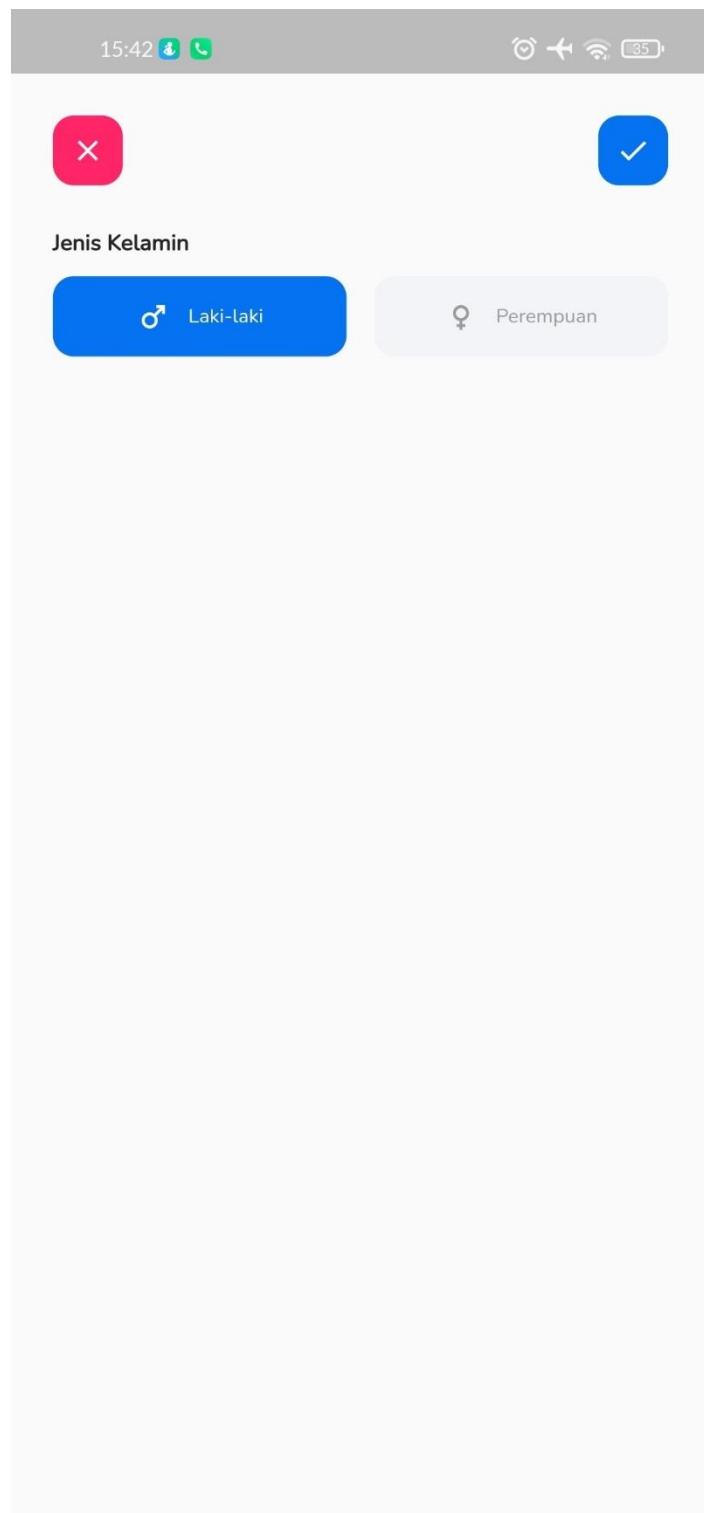
Gambar B. 18 Screenshot Halaman User

19. *Screenshot Halaman User Edit Nama*



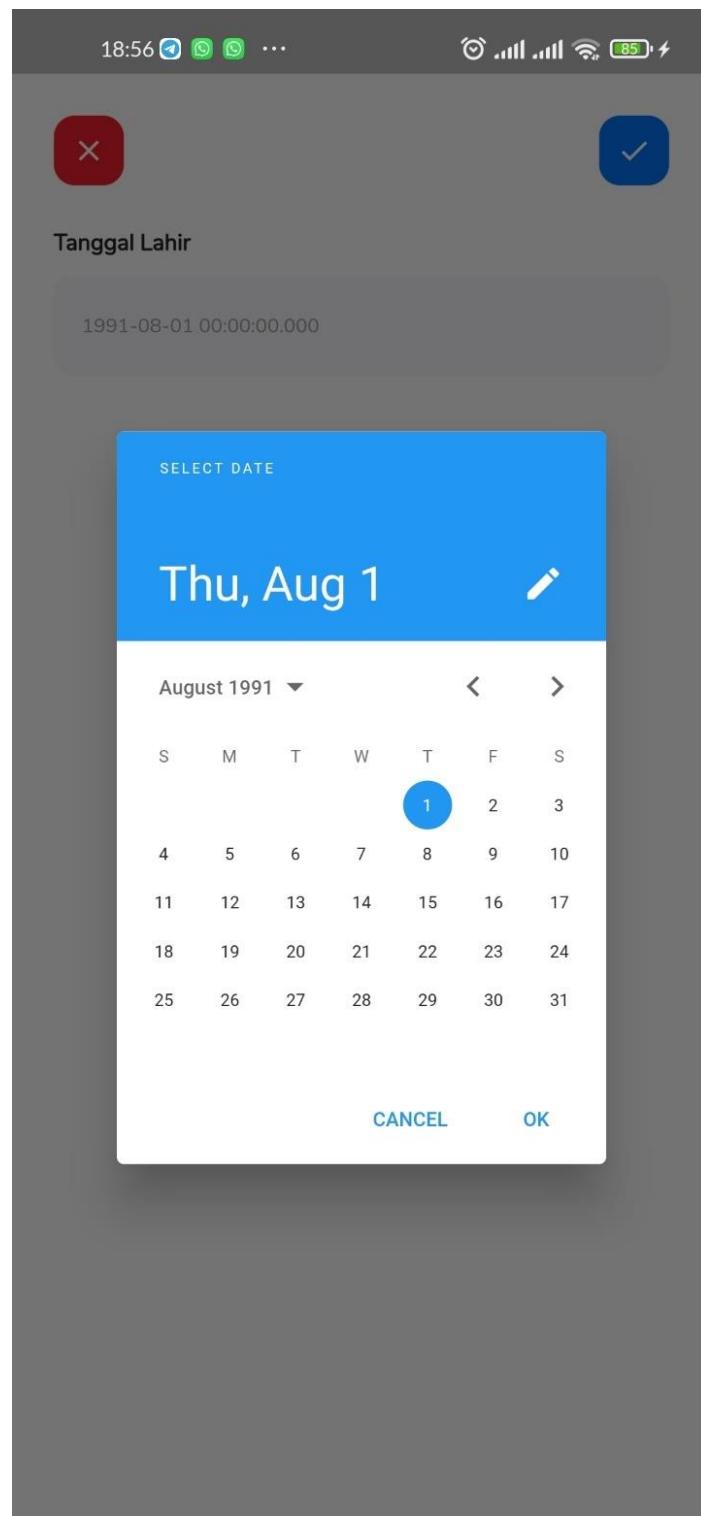
Gambar B. 19 Screenshot Halaman User Edit Nama

20. *Screenshot Halaman User Edit Jenis Kelamin*



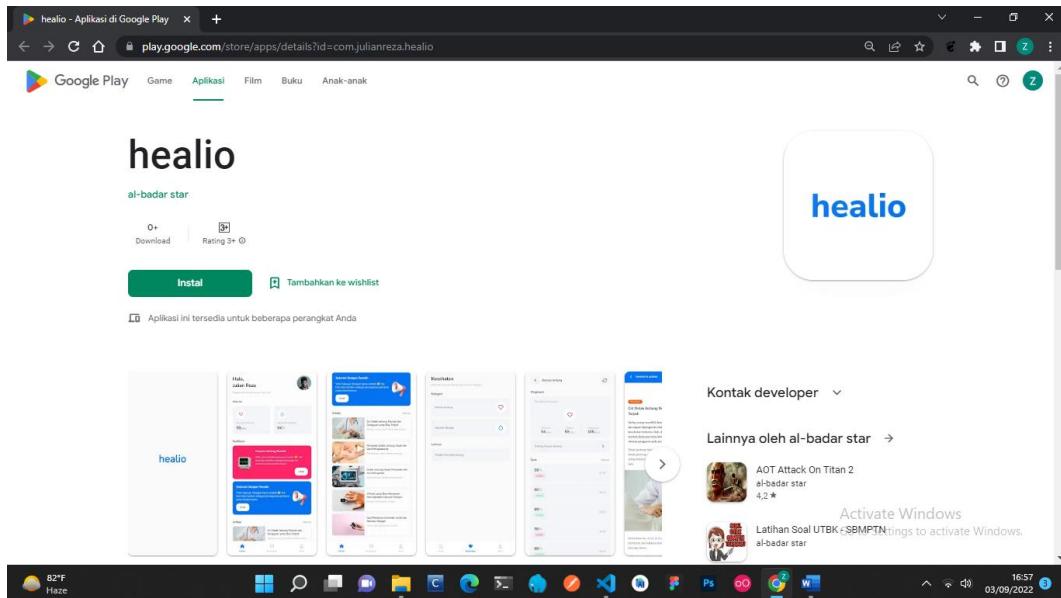
Gambar B. 20 Screenshot Halaman User Edit Jenis Kelamin

21. Screenshot Halaman User Edit Tanggal Lahir

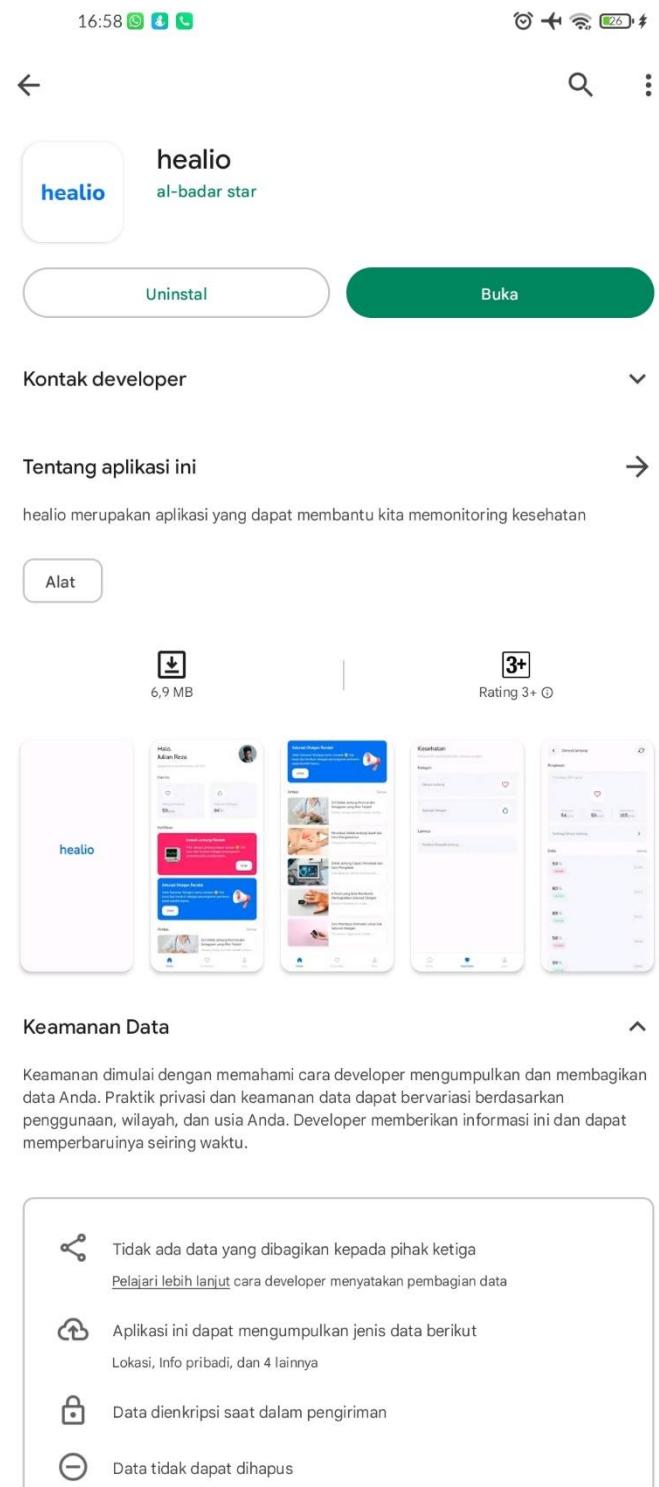


Gambar B. 21 Screenshot Halaman User Edit Tanggal Lahir

Lampiran C Hasil Review di Google Playstore



Gambar C. 1 Hasil Review di Playstore



Gambar C. 2 Hasil Review di Playstore

Lampiran D Listing Program

1. main.dart

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/article/article_bloc.dart';
import 'package:healio/blocs/auth/auth_bloc.dart';
import 'package:healio/blocs/health/health_bloc.dart';
import 'package:healio/blocs/heart_disease/heart_disease_bloc.dart';
import 'package:healio/blocs/page/page_cubit.dart';
import 'package:healio/blocs/user/user_bloc.dart';
import 'package:healio/pages/all_blood_oxygen_page.dart';
import 'package:healio/pages/all_heart_rate_page.dart';
import 'package:healio/pages/article_page.dart';
import 'package:healio/pages/blood_oxygen_information_page.dart';
import 'package:healio/pages/blood_oxygen_page.dart';
import 'package:healio/pages/heart_disease_check_page.dart';
import 'package:healio/pages/heart_rate_information_page.dart';
import 'package:healio/pages/heart_rate_page.dart';
import 'package:healio/pages/main_page.dart';
import 'package:healio/pages/sign_in_page.dart';
import 'package:healio/pages/sign_up_page.dart';
import 'package:healio/pages/splash_page.dart';

void main() async {
    WidgetsFlutterBinding.ensureInitialized();
    await Firebase.initializeApp();
    runApp(const MyApp());
}

class MyApp extends StatelessWidget {
    const MyApp({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        return MultiBlocProvider(
            providers: [
                BlocProvider(
                    create: (context) => PageCubit(),
                ),
                BlocProvider(
                    create: (context) => HealthBloc(),
                )
            ],
        );
    }
}
```

```

),
BlocProvider(
  create: (context) => ArticleBloc(),
),
BlocProvider(
  create: (context) => AuthBloc(),
),
BlocProvider(
  create: (context) => UserBloc(),
),
BlocProvider(
  create: (context) => HeartDiseaseBloc(),
),
],
child: MaterialApp(
  debugShowCheckedModeBanner: false,
  routes: {
    '/': (context) => const SplashPage(),
    '/main': (context) => const MainPage(),
    '/sign-in': (context) => const SignInPage(),
    '/sign-up': (context) => const SignUpPage(),
    '/article': (context) => const ArticlePage(),
    '/heart-rate': (context) => const HeartRatePage(),
    '/all-heart-rate': (context) => const AllHeartRatePage(),
    '/all-blood-oxygen': (context) => const
      AllBloodOxygenPage(),
    '/heart-rate-information': (context) =>
      const HeartRateInformationPage(),
    '/blood-oxygen': (context) => const BloodOxygenPage(),
    '/blood-oxygen-information': (context) =>
      const BloodOxygenInformationPage(),
    '/heart-disease-check': (context) => const
      HeartDiseaseCheckPage(),
  },
),
);
}
}
}

```

2. all_blood_oxygen_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/health/health_bloc.dart';

```

```

import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';
import 'package:healio/widget/health_data_item.dart';
import 'package:health/health.dart';

class AllBloodOxygenPage extends StatefulWidget {
  const AllBloodOxygenPage({Key? key}) : super(key: key);

  @override
  State<AllBloodOxygenPage> createState() =>
  _AllBloodOxygenPageState();
}

class _AllBloodOxygenPageState extends State<AllBloodOxygenPage> {
  @override
  void initState() {
    super.initState();
    context.read<HealthBloc>().add(HealthGetAllBloodOxygen());
  }

  getValueFromString(HealthValue healthValue) {
    List<String> value = healthValue.toString().split('.');
    return int.parse(value[0]);
  }

  @override
  Widget build(BuildContext context) {
    Widget header() {
      return const CustomAppBar(
        title: 'Saturasi Oksigen',
      );
    }

    Widget body() {
      return BlocBuilder<HealthBloc, HealthState>(
        builder: (context, state) {
          if (state is HealthAllBloodOxygenSuccess) {
            return Container(
              margin: EdgeInsets.only(
                left: defaultMargin,
                right: defaultMargin,
                bottom: defaultMargin,
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  Text('Saturasi Oksigen'),
                  Text(state.data.value.toString())
                ],
              ),
            );
          }
        },
      );
    }
  }
}

```

```

        children: [
          Text(
            'Data Semua Saturasi Oksigen',
            style: blackTextStyle.copyWith(
              fontSize: 16,
              fontWeight: bold,
            ),
          ),
        ],
        Column(
          children: state.allBloodOxygen
            .map((health) => HealthDataItem(
              health,
              isAll: true,
              category: 'blood_oxygen',
            ))
            .toList(),
        ),
      ],
    );
  }
  return Container();
},
);
}
}

return Scaffold(
  body: ListView(
    children: [
      header(),
      body(),
    ],
  ),
);
}
}
}

```

3. all_heart_rate_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:heilio/blocs/health/health_bloc.dart';

```

```
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';
import 'package:healio/widget/health_data_item.dart';
import 'package:health/health.dart';

class AllHeartRatePage extends StatefulWidget {
  const AllHeartRatePage({Key? key}) : super(key: key);

  @override
  State<AllHeartRatePage> createState() => _AllHeartRatePageState();
}

class _AllHeartRatePageState extends State<AllHeartRatePage> {
  @override
  void initState() {
    super.initState();
    context.read<HealthBloc>().add(HealthGetHeartRate());
    context.read<HealthBloc>().add(HealthGetAllHeartRate());
  }

  getValueFromString(HealthValue healthValue) {
    List<String> value = healthValue.toString().split('.');
    return int.parse(value[0]);
  }

  @override
  Widget build(BuildContext context) {
    Widget header() {
      return const CustomAppBar(
        title: 'Denyut Jantung',
      );
    }

    Widget body() {
      return BlocBuilder<HealthBloc, HealthState>(
        builder: (context, state) {
          if (state is HealthAllHeartRateSuccess) {
            return Container(
              margin: EdgeInsets.only(
                left: defaultMargin,
                right: defaultMargin,
                bottom: defaultMargin,
              ),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
```

```

        children: [
          Text(
            'Data Semua Denyut Jantung',
            style: blackTextStyle.copyWith(
              fontSize: 16,
              fontWeight: bold,
            ),
          ),
        ),
        Column(
          children: state.allHeartRate
            .map((health) => HealthDataItem(
              health,
              isAll: true,
            ))
            .toList(),
        ),
      ],
    );
  }
  return Container();
},
);
}
}

return Scaffold(
  body: ListView(
    children: [
      header(),
      body(),
    ],
  ),
);
}
}
}

```

4. article_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:heilio/models/article_model.dart';
import 'package:heilio/pages/webview_page.dart';

```

```
import 'package:healio/widget/article_card.dart';

import '../blocs/article/article_bloc.dart';
import '../shared/theme.dart';

class ArticlePage extends StatelessWidget {
  const ArticlePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Widget header() {
      return Container(
        padding: const EdgeInsets.all(24),
        child: Row(
          children: [
            GestureDetector(
              onTap: () {
                Navigator.pop(context);
              },
              child: Container(
                padding: const EdgeInsets.all(8),
                decoration: BoxDecoration(
                  color: kLightGrayColor,
                  borderRadius: BorderRadius.circular(
                    12,
                  ),
                ),
                margin: const EdgeInsets.only(right: 10),
                child: const Icon(
                  Icons.chevron_left,
                  size: 24,
                ),
              ),
            ),
            Text(
              'Artikel',
              style: blackTextStyle.copyWith(
                fontSize: 16,
                fontWeight: bold,
              ),
            ),
          ],
        );
    }
}
```

```

Widget article() {
    return Container(
        margin: const EdgeInsets.only(
            left: 24,
            right: 24,
            bottom: 24,
        ),
        child: Column(
            children: listArticle.map((article) {
                return ArticleCard(article);
            }).toList(),
        ),
    );
}

return Scaffold(
    body: ListView(
        children: [
            header(),
            article(),
        ],
    ),
);
}
}
}

```

5. blood_oxygen_information_page.dart

```

import 'package:flutter/material.dart';
import 'package:healio/widget/custom_app_bar.dart';

import '../shared/theme.dart';

class BloodOxygenInformationPage extends StatelessWidget {
    const BloodOxygenInformationPage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        Widget header() {
            return const CustomAppBar(
                title: 'Tentang Saturasi Oksigen',

```

```

    );
}

Widget content() {
    return Container(
        margin: EdgeInsets.only(
            right: defaultMargin,
            left: defaultMargin,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                ClipRRect(
                    borderRadius: BorderRadius.circular(12),
                    child: Image.asset(
                        'assets/bo_thumbnail.jpg',
                    ),
                ),
                SizedBox(
                    height: defaultMargin,
                ),
                Text(
                    'Saturasi Oksigen',
                    style: blackTextStyle.copyWith(
                        fontSize: 18,
                        fontWeight: semiBold,
                    ),
                ),
                const SizedBox(
                    height: 10,
                ),
                Text(
                    'Saturasi oksigen merupakan nilai yang menunjukkan kadar oksigen di dalam darah. Nilai ini sangat berpengaruh terhadap berbagai fungsi organ dan jaringan tubuh. Pengukuran nilai saturasi oksigen dapat dilakukan dengan 2 cara, yakni dengan analisis gas darah (AGD) atau menggunakan alat oximeter.\n\nHasil pengukuran saturasi oksigen yang dilakukan dengan analisis gas darah ditunjukkan dengan istilah PaO2 (tekanan parsial oksigen). Sementara itu, hasil pengukuran saturasi oksigen dengan menggunakan oximeter ditunjukkan dengan istilah SpO2.',
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                    ),
                ),
            ],
        ),
    );
}

```

```

        SizedBox(
            height: defaultMargin,
        ),
        Text(
            'Saturasi Oksigen Normal',
            style: blackTextStyle.copyWith(
                fontSize: 18,
                fontWeight: bold,
            ),
        ),
        const SizedBox(
            height: 10,
        ),
        Text(
            'Berikut adalah nilai saturasi oksigen normal pada orang dengan kondisi paru-paru yang sehat atau tidak memiliki kondisi medis tertentu:',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
        Text(
            '          • Analisis gas darah (PaO2): 80–100 mmHg\n
• Oximeter (SpO2): 95–100%',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
        SizedBox(height: defaultMargin),
        Text(
            'Saturasi Oksigen Rendah',
            style: blackTextStyle.copyWith(
                fontSize: 18,
                fontWeight: bold,
            ),
        ),
        const SizedBox(
            height: 10,
        ),
        Text(
            'Berikut adalah kriteria nilai saturasi oksigen rendah atau di bawah normal:',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
    
```

```

),
Text(
      • Analisis gas darah (PaO2): dibawah 80 mmHg\n
• Oximeter (SpO2): dibawah 95%\n\nOrang yang memiliki saturasi oksigen rendah atau hipoksemia bisa merasakan berbagai gejala, seperti nyeri dada, sesak napas, batuk, sakit kepala, detak jantung cepat, kebingungan, dan kulit membiru.\n\nKendati demikian, orang yang mengalami hipoksemia juga bisa tidak merasakan gejala apa pun. Kondisi ini yang disebut dengan happy hypoxia ini bisa terjadi pasien COVID-19.\n\nHipoksemia, baik yang menimbulkan gejala maupun tidak, bisa mengganggu kerja organ dan jaringan tubuh. Bila dibiarkan, hal ini dapat menyebabkan kerusakan pada organ vital, seperti jantung, otak, dan ginjal, dan berisiko menyebabkan komplikasi yang berbahaya.',  

      style: blackTextStyle.copyWith(  

        fontSize: 16,  

      ),  

    ),  

    SizedBox(  

      height: defaultMargin,  

    ),  

    const Divider(),  

  ],  

),
);
}  

Widget disclaimer() {  

  return Container(  

    margin: EdgeInsets.all(defaultMargin),  

    child: Column(  

      mainAxisAlignment: CrossAxisAlignment.start,  

      children: [  

        Row(  

          mainAxisAlignment: CrossAxisAlignment.center,  

          children: [  

            Icon(  

              Icons.warning,  

              size: 16,  

              color: kRedColor,  

            ),  

            const SizedBox(  

              width: 5,  

            ),  

            Text(

```

```

        'Disclaimer',
        style: blackTextStyle.copyWith(
            fontSize: 18,
            fontWeight: bold,
        ),
    ),
],
),
const SizedBox(
    height: 10,
),
Text(
    'Harap diingat bahwa pemantauan saturasi oksigen hanya
untuk kebugaran dan kesehatan, bukan untuk diagnosis atau perawatan
kondisi medis apapun. Jika Anda memiliki kekhawatiran tentang
jantung Anda, pastikan untuk berkonsultasi dengan profesional
medis.',
    style: blackTextStyle.copyWith(
        fontSize: 16,
    ),
),
],
),
);
},
);
}
}

return Scaffold(
body: ListView(
children: [
header(),
content(),
disclaimer(),
],
),
);
}
}

```

6. blood_oxygen_page.dart

```

import 'dart:developer';
import 'dart:math' as math;

```

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/health/health_bloc.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';
import 'package:healio/widget/health_data_item.dart';
import 'package:healio/widget/more_button.dart';
import 'package:health/health.dart';

import '../blocs/auth/auth_bloc.dart';
import '../models/health_model.dart';
import '../services/health_service.dart';
import '../services/push_notification_service.dart';
import '../shared/shared_methods.dart';

class BloodOxygenPage extends StatefulWidget {
  const BloodOxygenPage({Key? key}) : super(key: key);

  @override
  State<BloodOxygenPage> createState() => _BloodOxygenPageState();
}

late HealthModel boFromFirebase;

List<HealthDataPoint> dataBloodOxygen = [];

int age = 0;

String category = '';

class _BloodOxygenPageState extends State<BloodOxygenPage> {
  @override
  void initState() {
    super.initState();
    getDataFromFirebase();
    fetchData();
    getAge();
  }

  void fetchData() {
    context.read<HealthBloc>().add(HealthGetBloodOxygen());
    final healthState = context.read<HealthBloc>().state;
    if (healthState is HealthBloodOxygenSuccess) {
      dataBloodOxygen = healthState.bloodOxygen;
    }
  }
}

```

```

        }

    }

    void getAge() {
        final authState = context.read<AuthBloc>().state;
        if (authState is AuthSuccess) {
            age =
            calculateAge(DateTime.parse(authState.user.dateOfBirth));
        }
    }

    void getDataFromFirebase() async {
        boFromFirebase =
        await HealthService().getBloodOxygenFromFirebase();
    }

    void updateBloodOxygen() async {
        if (boFromFirebase.value ==
            getValueFromHealthValue(dataBloodOxygen.first.value) &&
            boFromFirebase.dateFrom ==
            dataBloodOxygen.first.dateFrom.toString()) {
            log('Sama');
            log(age.toString());
        } else {
            log('beda');
            await HealthService().updateBloodOxygenFromFirebase(
                getValueFromHealthValue(dataBloodOxygen.first.value),
                dataBloodOxygen.first.dateFrom.toString());
            category = setCategory(
                getValueFromHealthValue(dataBloodOxygen.first.value),
                'blood_oxygen',
                age,
            );
            log(category);
            if (category == 'rendah') {
                customShowSnackBar(context, 'Saturasi oksigen kamu rendah');
                await PushNotificationService().pushNotification(
                    'Saturasi Oksigen',
                    ${getValueFromHealthValue(dataBloodOxygen.first.value)},
                    'Saturasi Oksigen kamu dibawah nilai normal nih :(.',
                );
            }
        }
    }
}

```

```

syncData() async {
    fetchData();
    getDataFromFirebase();
    updateBloodOxygen();
}

@Override
Widget build(BuildContext context) {
    Widget header() {
        return CustomAppBar(
            title: 'Saturasi Oksigen',
            action: GestureDetector(
                onTap: () {
                    syncData();
                },
                child: Container(
                    padding: const EdgeInsets.all(10),
                    decoration: BoxDecoration(
                        color: kLightGrayColor,
                        borderRadius: BorderRadius.circular(
                            defaultRadius,
                        ),
                    ),
                    child: Image.asset(
                        'assets/fi_refresh-cw.png',
                        height: 20,
                    ),
                ),
            ),
        );
    }

    Widget report() {
        return BlocBuilder<HealthBloc, HealthState>(
            builder: (context, state) {
                if (state is HealthBloodOxygenSuccess) {
                    List<int> listValue = state.bloodOxygen
                        .map((e) => getValueFromHealthValue(e.value))
                        .toList();

                    return Container(
                        margin: EdgeInsets.symmetric(
                            horizontal: defaultMargin,
                        ),
                        child: Column(

```

```
crossAxisAlignment: CrossAxisAlignment.start,
children: [
    Text(
        'Ringkasan',
        style: blackTextStyle.copyWith(
            fontSize: 16,
            fontWeight: bold,
        ),
    ),
    const SizedBox(height: 10),
    Container(
        padding: const EdgeInsets.all(16),
        decoration: BoxDecoration(
            color: kLightGrayColor,
            borderRadius: BorderRadius.circular(
                defaultRadius,
            ),
        ),
        child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
                Text(
                    'Rabu, 12 Juli',
                    style: grayTextStyle,
                ),
                const SizedBox(height: 24),
                Column(
                    children: [
                        Container(
                            height: 56,
                            width: 56,
                            decoration: BoxDecoration(
                                color: kWhiteColor,
                                shape: BoxShape.circle,
                            ),
                            padding: const EdgeInsets.all(16),
                            child: Image.asset(
                                'assets/fi_droplet.png',
                                color: kPrimaryColor,
                            ),
                        ),
                        const SizedBox(height: 24),
                        Row(
                            children: [
                                Expanded(
```

```

        child: Container(
          decoration: BoxDecoration(
            border: Border(
              right: BorderSide(
                width: 1,
                color: kGrayColor.withOpacity(0.3),
              ),
            ),
          ),
        ),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(
            'Minimum:',
            style: grayTextStyle.copyWith(
              fontSize: 12,
            ),
          ),
          Wrap(
            crossAxisAlignment: WrapCrossAlignment.end,
            children: [
              Text(
                '${state.bloodOxygen.isEmpty ? 0 : listValue.reduce(math.min)}',
                style: blackTextStyle.copyWith(
                  fontSize: 20,
                  fontWeight: FontWeight.bold,
                ),
              ),
              Text(
                '%',
                style: grayTextStyle.copyWith(
                  fontSize: 20,
                ),
              ),
            ],
          ),
        ],
      ),
    ],
  ),

```

```
        ),
        ),
        ),
        Expanded(
            child: Container(
                decoration: BoxDecoration(
                    border: Border(
                        right: BorderSide(
                            width: 1,
                            color: kGrayColor.withOpacity(0.3),
                    ),
                    ),
                    ),
                    ),
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                            Text(
                                'Terbaru:',
                                style: grayTextStyle.copyWith(
                                    fontSize: 12,
                                ),
                            ),
                            Wrap(
                                crossAxisAlignment: WrapCrossAlignment.end,
                                children: [
                                    Text(
                                        '${state.bloodOxygen.isEmpty ? 0 : getValueFromHealthValue(state.bloodOxygen.first.value)}',
                                        style: blackTextStyle.copyWith(
                                            fontSize: 20,
                                            fontWeight: FontWeight.bold,
                                        ),
                                    ),
                                    Text(
                                        '%',
                                        style: grayTextStyle.copyWith(
                                            fontSize: 20,
                                        ),
                                    ),
                                ],
                            ),
                        ],
                    ),
                ),
            ),
        ),
    ),
)
```

```
            ),  
            ),  
            ],  
            ),  
            ],  
            ),  
            ),  
            ),  
            ),  
            Expanded(  
              child: Column(  
                mainAxisAlignment:  
                  CrossAxisAlignment.center,  
                children: [  
                  Text(  
                    'Maksimum:',  
                    style:  
                      grayTextStyle.copyWith(  
                        fontSize: 12,  
                      ),  
                      ),  
                      Wrap(  
                        mainAxisAlignment:  
                          WrapCrossAlignment.end,  
                        children: [  
                          Text(  
  
                            '${state.bloodOxygen.isEmpty ? 0 : listValue.reduce(math.max)}',  
                            style:  
                              blackTextStyle.copyWith(  
                                fontSize: 20,  
                                fontWeight: bold,  
                              ),  
                              ),  
                              Text(  
                                '%',  
                                style:  
                                  grayTextStyle.copyWith(  
                                    fontSize: 20,  
                                    ),  
                                    ),  
                                    ],  
                                    ),  
                                    ],  
                                    ),  
                                    ),  
                                    ),  
                                    ),  
          ),
```

```

        ],
        ),
        ],
        ),
        ],
        ),
        ],
        ),
        ],
        ),
        );
    }
    return Container();
},
);
}
}

Widget more() {
return const MoreButton(
text: 'Tentang Saturasi Oksigen',
route: '/blood-oxygen-information',
);
}

Widget history() {
return BlocBuilder<HealthBloc, HealthState>(
builder: (context, state) {
if (state is HealthBloodOxygenSuccess) {
return Container(
margin: EdgeInsets.all(
defaultMargin,
),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Row(
children: [
Text(
'Data',
style: blackTextStyle.copyWith(
fontSize: 16,
fontWeight: bold,
),
),
const Spacer(),
GestureDetector(

```

```

        onTap: () {
            Navigator.pushNamed(context, '/all-blood-oxygen');
        },
        child: Text(
            'Semua',
            style: grayTextStyle.copyWith(
                fontSize: 14,
                fontWeight: bold,
            ),
        ),
    ],
),
state.bloodOxygen.isEmpty
? Center(
    child: Container(
        margin: const EdgeInsets.only(
            top: 56,
        ),
    ),
    padding: const
EdgeInsets.symmetric(horizontal: 36),
    child: Column(
        children: [
            Image.asset(
                'assets/document.png',
                height: 100,
            ),
            const SizedBox(height: 10),
            Text(
                'Data Kosong',
                style: blackTextStyle.copyWith(
                    fontSize: 20,
                    fontWeight: bold,
                ),
            ),
            const SizedBox(height: 5),
            Text(
                'Data saturasi oksigen kamu hari ini masih kosong nih, yuk lakukan pengecekan pada smartwatch kamu.',
                style: grayTextStyle.copyWith(),
                textAlign: TextAlign.center,
            ),
        ],
),
),

```

```

        ),
    )
: Column(
    children: state.bloodOxygen
        .map(
            (health) => HealthDataItem(
                health,
                category: 'blood_oxygen',
            ),
        )
        .toList(),
),
],
),
);
}
return Container();
},
);
}
}

return Scaffold(
body: ListView(
children: [
header(),
report(),
more(),
history(),
],
),
);
}
}
}

```

7. edit_date_of_birth_page.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/shared/shared_methods.dart';
import 'package:healio/shared/theme.dart';

```

```
import '../blocs/auth/auth_bloc.dart';
import '../blocs/user/user_bloc.dart';

class EditDateOfBirthPage extends StatefulWidget {
    const EditDateOfBirthPage({required this.date, Key? key}) : super(key: key);

    final String date;

    @override
        State<EditDateOfBirthPage> createState() =>
    _EditDateOfBirthPageState();
}

late DateTime initialValue;
late String selectedDate;

class _EditDateOfBirthPageState extends State<EditDateOfBirthPage> {
    User? user = FirebaseAuth.instance.currentUser;
    bool validate() {
        if (selectedDate.isEmpty) {
            return false;
        }
        return true;
    }

    @override
    void initState() {
        super.initState();
        initialValue = DateTime.parse(widget.date);
        selectedDate = widget.date;
    }

    @override
    Widget build(BuildContext context) {
        Widget header() {
            return Container(
                margin: EdgeInsets.all(
                    defaultMargin,
                ),
                child: Row(
                    children: [
                        GestureDetector(
                            onTap: () {
                                Navigator.pop(context);
                            }
                        )
                    ],
                )
            );
        }
    }
}
```

```
        },
        child: Container(
            padding: const EdgeInsets.all(10),
            decoration: BoxDecoration(
                color: kRedColor,
                borderRadius: BorderRadius.circular(
                    defaultRadius,
                ),
            ),
            child: Icon(
                Icons.close,
                size: 20,
                color: kWhiteColor,
            ),
        ),
    ),
    const Spacer(),
    BlocConsumer<UserBloc, UserState>(
        listener: (context, state) {
            if (state is UserSuccess) {

                context.read<AuthBloc>().add(AuthGetCurrentUser(user!.uid));
                Navigator.pushNamed(context, '/main');
            }
        },
        builder: (context, state) {
            if (state is UserLoading) {
                return Container(
                    padding: const EdgeInsets.all(10),
                    decoration: BoxDecoration(
                        color: kPrimaryColor,
                        borderRadius: BorderRadius.circular(
                            defaultRadius,
                        ),
                    ),
                    child: SizedBox(
                        height: 20,
                        width: 20,
                        child: CircularProgressIndicator(
                            color: kWhiteColor,
                        ),
                    ),
                );
            }
        });
    return GestureDetector(
```

```

        onTap: () {
            if (validate()) {
                context.read<UserBloc>().add(
                    UserUpdate('date_of_birth', selectedDate,
user!.uid));
            } else {
                customShowSnackBar(
                    context, 'Field nama tidak boleh kosong');
            }
        },
        child: Container(
            padding: const EdgeInsets.all(10),
            decoration: BoxDecoration(
                color: kPrimaryColor,
                borderRadius: BorderRadius.circular(
                    defaultRadius,
                ),
            ),
            child: Icon(
                Icons.check,
                size: 20,
                color: kWhiteColor,
            ),
        ),
    ),
),
),
),
],
),
);
}
}

Widget body() {
return Container(
margin: EdgeInsets.symmetric(
    horizontal: defaultMargin,
),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
    'Tanggal Lahir',
    style: blackTextStyle.copyWith(
        fontSize: 16,
        fontWeight: bold,

```

```
        ),
    ),
),
const SizedBox(height: 10),
GestureDetector(
    onTap: () {
        showDatePicker(
            context: context,
            initialDate: initialDate,
            firstDate: DateTime(1950),
            lastDate: DateTime(2050),
        ).then((value) {
            if (value != null) {
                setState(() {
                    selectedDate = value.toString();
                });
            }
        });
    },
),
child: Container(
    decoration: BoxDecoration(
        color: kLightGrayColor,
        borderRadius: BorderRadius.circular(
            defaultRadius,
        ),
    ),
),
padding: const EdgeInsets.symmetric(
    horizontal: 16,
    vertical: 10,
),
height: 57,
width: double.infinity,
child: Row(
    children: [
        Text(
            selectedDate,
            style: grayTextStyle,
        ),
    ],
),
),
),
],
),
);
}
```

```

        return Scaffold(
            body: ListView(
                children: [
                    header(),
                    body(),
                ],
            ),
        );
    }
}

```

8. edit_gender_page.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/shared/theme.dart';

import '../blocs/auth/auth_bloc.dart';
import '../blocs/user/user_bloc.dart';

class EditGenderPage extends StatefulWidget {
    const EditGenderPage({Key? key}) : super(key: key);

    @override
    State<EditGenderPage> createState() => _EditGenderPageState();
}

class _EditGenderPageState extends State<EditGenderPage> {
    String gender = 'male';
    User? user = FirebaseAuth.instance.currentUser;
    @override
    Widget build(BuildContext context) {
        Widget header() {
            return Container(
                margin: EdgeInsets.all(
                    defaultMargin,
                ),
                child: Row(
                    children: [
                        GestureDetector(

```

```

        onTap: () {
            Navigator.pop(context);
        },
        child: Container(
            padding: const EdgeInsets.all(10),
            decoration: BoxDecoration(
                color: kRedColor,
                borderRadius: BorderRadius.circular(
                    defaultRadius,
                ),
            ),
            child: Icon(
                Icons.close,
                size: 20,
                color: kWhiteColor,
            ),
        ),
    ),
    const Spacer(),
    BlocConsumer<UserBloc, UserState>(
        listener: (context, state) {
            if (state is UserSuccess) {

context.read<AuthBloc>().add(AuthGetCurrentUser(user!.uid));
                Navigator.pushNamed(context, '/main');
            }
        },
        builder: (context, state) {
            if (state is UserLoading) {
                return Container(
                    padding: const EdgeInsets.all(10),
                    decoration: BoxDecoration(
                        color: kPrimaryColor,
                        borderRadius: BorderRadius.circular(
                            defaultRadius,
                        ),
                    ),
                    child: SizedBox(
                        height: 20,
                        width: 20,
                        child: CircularProgressIndicator(
                            color: kWhiteColor,
                        ),
                    ),
                );
            };
        },
    );

```

```
        }
        return GestureDetector(
            onTap: () {
                context
                    .read<UserBloc>()
                    .add(UserUpdate('gender', gender,
user!.uid));
            },
            child: Container(
                padding: const EdgeInsets.all(10),
                decoration: BoxDecoration(
                    color: kPrimaryColor,
                    borderRadius: BorderRadius.circular(
                        defaultRadius,
                    ),
                ),
                child: Icon(
                    Icons.check,
                    size: 20,
                    color: kWhiteColor,
                ),
            ),
        );
    },
),
],
),
);
}
}

Widget body() {
return Container(
margin: EdgeInsets.symmetric(
    horizontal: defaultMargin,
),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
    'Jenis Kelamin',
    style: blackTextStyle.copyWith(
        fontSize: 16,
        fontWeight: bold,
    ),
),
],
)
```

```

const SizedBox(height: 10),
Row(
  children: [
    Expanded(
      child: SizedBox(
        height: 46,
        child: ElevatedButton(
          onPressed: () {
            setState(() {
              gender = 'male';
            });
          },
          style: ElevatedButton.styleFrom(
            elevation: 0,
            primary:
              gender == 'male' ? kPrimaryColor :
kLightGrayColor,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(
                defaultRadius,
              ),
            ),
            ),
            ),
            child: Wrap(
              crossAxisAlignment:
WrapCrossAxisAlignment.center,
              children: [
                Icon(
                  Icons.male,
                  size: 20,
                  color: gender == 'male' ? kWhiteColor :
kGrayColor,
                ),
                const SizedBox(
                  width: 10,
                ),
                Text(
                  'Laki-laki',
                  style: whiteTextStyle.copyWith(
                    color:
                      gender == 'male' ? kWhiteColor :
kGrayColor,
                  ),
                ),
              ],
            ),
          ],
        ),
      ],
    ),
  ],
)

```

```
        ),
        ),
        ),
        ),
        const SizedBox(width: 16),
        Expanded(
            child: SizedBox(
                height: 46,
                child: ElevatedButton(
                    onPressed: () {
                        setState(() {
                            gender = 'female';
                        });
                    },
                    style: ElevatedButton.styleFrom(
                        elevation: 0,
                        primary:
                            gender == 'female' ? kRedColor : kLightGrayColor,
                        shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(
                                defaultRadius,
                            ),
                            ),
                            ),
                            ),
                            child: Wrap(
                                crossAxisAlignment:
WrapCrossAxisAlignment.center,
                                children: [
                                    Icon(
                                        Icons.female,
                                        size: 20,
                                        color:
                                            gender == 'female' ? kWhiteColor : kGrayColor,
                                    ),
                                    const SizedBox(
                                        width: 10,
                                    ),
                                    Text(
                                        'Perempuan',
                                        style: whiteTextStyle.copyWith(
                                            color:
                                                gender == 'female' ? kWhiteColor : kGrayColor,
                                            
```

```

        ),
        ),
        ],
        ),
        ),
        ),
        ),
        ],
        ),
        ],
        ),
        );
    }

    return Scaffold(
        body: ListView(
            children: [
                header(),
                body(),
                ],
            ),
        );
    }
}

```

9. edit_name_page.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/user/user_bloc.dart';

import '../blocs/auth/auth_bloc.dart';
import '../shared/shared_methods.dart';
import '../shared/theme.dart';

class EditNamePage extends StatelessWidget {
    const EditNamePage({required this.name, Key? key}) : super(key: key);

    final String name;
}

```

```

@Override
Widget build(BuildContext context) {
    final nameController = TextEditingController(text: name);
    User? user = FirebaseAuth.instance.currentUser;

    validate() {
        if (nameController.text.isEmpty) {
            return false;
        }
        return true;
    }

    Widget header() {
        return Container(
            margin: EdgeInsets.all(
                defaultMargin,
            ),
            child: Row(
                children: [
                    GestureDetector(
                        onTap: () {
                            Navigator.pop(context);
                        },
                        child: Container(
                            padding: const EdgeInsets.all(10),
                            decoration: BoxDecoration(
                                color: kRedColor,
                                borderRadius: BorderRadius.circular(
                                    defaultRadius,
                                ),
                            ),
                            child: Icon(
                                Icons.close,
                                size: 20,
                                color: kWhiteColor,
                            ),
                        ),
                    ),
                    const Spacer(),
                    BlocConsumer<AuthBloc, UserState>(
                        listener: (context, state) {
                            if (state is UserSuccess) {

context.read<AuthBloc>().add(AuthGetCurrentUser(user!.uid));
                            Navigator.pushNamed(context, '/main');

```

```
        },
    },
    builder: (context, state) {
        if (state is UserLoading) {
            return Container(
                padding: const EdgeInsets.all(10),
                decoration: BoxDecoration(
                    color: kPrimaryColor,
                    borderRadius: BorderRadius.circular(
                        defaultRadius,
                    ),
                ),
                child: SizedBox(
                    height: 20,
                    width: 20,
                    child: CircularProgressIndicator(
                        color: kWhiteColor,
                    ),
                ),
            );
        }
        return GestureDetector(
            onTap: () {
                if (validate()) {
                    context.read<UserBloc>().add(
                        UserUpdate('name', nameController.text,
user!.uid));
                } else {
                    customShowSnackBar(
                        context, 'Field nama tidak boleh kosong');
                }
            },
            child: Container(
                padding: const EdgeInsets.all(10),
                decoration: BoxDecoration(
                    color: kPrimaryColor,
                    borderRadius: BorderRadius.circular(
                        defaultRadius,
                    ),
                ),
                child: Icon(
                    Icons.check,
                    size: 20,
                    color: kWhiteColor,
                ),
            ),
        );
    },
},
```

```
        ),
        ),
    },
),
],
),
);
}

Widget body() {
return Container(
margin: EdgeInsets.symmetric(
    horizontal: defaultMargin,
),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
    'Nama',
    style: blackTextStyle.copyWith(
        fontSize: 16,
        fontWeight: bold,
    ),
),
const SizedBox(height: 10),
Container(
decoration: BoxDecoration(
    color: kLightGrayColor,
    borderRadius: BorderRadius.circular(
        defaultRadius,
    ),
),
padding: const EdgeInsets.symmetric(
    horizontal: 16,
    vertical: 10,
),
height: 57,
child: Center(
    child: TextFormField(
        controller: nameController,
        style: blackTextStyle,
        decoration: InputDecoration.collapsed(
            hintText: 'Nama',
            hintStyle: grayTextStyle,
        ),
    ),
)
],
```

```

        ),
        ),
        ],
),
);
}

return Scaffold(
  body: ListView(
    children: [
      header(),
      body(),
    ],
),
);
}
}

```

10. health_page.dart

```

import 'package:flutter/material.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/health_category_tile.dart';

class HealthPage extends StatelessWidget {
  const HealthPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Widget header() {
      return Container(
        margin: EdgeInsets.all(defaultMargin),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Kesehatan',
              style: blackTextStyle.copyWith(
                fontSize: 26,
                fontWeight: bold,
            ),
        
```

```
        ),
        const SizedBox(height: 5),
        Text(
            'Meliputi denyut jantung dan saturasi oksigen',
            style: grayTextStyle,
        ),
    ],
),
);
}

Widget body() {
    return Container(
        margin: EdgeInsets.symmetric(
            horizontal: defaultMargin,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Text(
                    'Kategori',
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                        fontWeight: bold,
                    ),
                ),
                const SizedBox(height: 10),
                const HealthCategoryTile(
                    healthCategory: HealthCategory.heartrate,
                ),
                const SizedBox(height: 10),
                const HealthCategoryTile(
                    healthCategory: HealthCategory.bloodoxygen,
                ),
                const SizedBox(height: 24),
                Text(
                    'Lainnya',
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                        fontWeight: bold,
                    ),
                ),
            ],
        ),
        GestureDetector(
            onTap: () {
                Navigator.pushNamed(context, '/heart-disease-')
            }
        )
    );
}
```

```

        check');
    },
    child: Container(
        decoration: BoxDecoration(
            color: kLightGrayColor,
            borderRadius: BorderRadius.circular(12),
        ),
        margin: const EdgeInsets.only(top: 10),
        padding: const EdgeInsets.all(16),
        height: 57,
        child: Row(
            children: [
                Text(
                    'Prediksi Penyakit Jantung',
                    style: blackTextStyle,
                ),
                ],
            ),
        ),
        ),
        ],
    ),
);
}

return Scaffold(
    body: ListView(
        children: [
            header(),
            body(),
            ],
        ),
    );
}
}

```

11. heart_disease_check_page.dart

```

import 'dart:developer';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';

```

```

import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/models/heart_disease_form_model.dart';
import 'package:healio/shared/shared_methods.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';
import 'package:healio/widget/custom_button.dart';

import '../blocs/auth/auth_bloc.dart';
import '../blocs/health/health_bloc.dart';
import '../blocs/heart_disease/heart_disease_bloc.dart';
import '../widget/custom_text_form_field.dart';
import 'heart_disease_result_page.dart';

class HeartDiseaseCheckPage extends StatefulWidget {
  const HeartDiseaseCheckPage({Key? key}) : super(key: key);

  @override
  State<HeartDiseaseCheckPage> createState() =>
  _HeartDiseaseCheckPageState();
}

final ageController = TextEditingController(text: '');
final heartRateController = TextEditingController(text: '');
final cigarettesController = TextEditingController(text: '');
final glucoseController = TextEditingController(text: '');
final diabetesController = TextEditingController(text: '');
final cholesterolController = TextEditingController(text: '');
final systolicController = TextEditingController(text: '');
final diastolicController = TextEditingController(text: '');
String gender = 'male';

class _HeartDiseaseCheckPageState extends State<HeartDiseaseCheckPage> {
  @override
  void initState() {
    super.initState();
    fetchData();

    final healthState = context.read<HealthBloc>().state;
    if (healthState is HealthHeartRateSuccess) {
      heartRateController.text =
        getValueFromHealthValue(healthState.heartRate.first.value).toString();
    }
}

```

```
final authState = context.read<AuthBloc>().state;
if (authState is AuthSuccess) {
    ageController.text =
        calculateAge(DateTime.parse(authState.user.dateOfBirth)).toString();
}
}

fetchData() {
    context.read<HealthBloc>().add(HealthGetHeartRate());
}

@Override
Widget build(BuildContext context) {
    bool validate() {
        if (ageController.text.isEmpty ||
            heartRateController.text.isEmpty ||
            cigarettesController.text.isEmpty ||
            glucoseController.text.isEmpty ||
            diabetesController.text.isEmpty ||
            cholesterolController.text.isEmpty ||
            systolicController.text.isEmpty ||
            diastolicController.text.isEmpty) {
            return false;
        }
        return true;
    }

    Widget appBar() {
        return const CustomAppBar(title: 'Prediksi Penyakit Jantung');
    }

    Widget body() {
        return Container(
            margin: EdgeInsets.only(
                left: defaultMargin,
                right: defaultMargin,
                bottom: defaultMargin,
            ),
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                    CustomTextFormField(
                        label: 'Umur',
```

```

        controller: ageController,
    ),
    const SizedBox(height: 16),
    Text(
        'Jenis Kelamin',
        style: blackTextStyle.copyWith(
            fontSize: 16,
            fontWeight: bold,
        ),
    ),
    const SizedBox(height: 10),
    Row(
        children: [
            Expanded(
                child: SizedBox(
                    height: 46,
                    child: ElevatedButton(
                        onPressed: () {
                            setState(() {
                                gender = 'male';
                            });
                        },
                        style: ElevatedButton.styleFrom(
                            elevation: 0,
                            primary:
                                gender == 'male' ? kPrimaryColor :
kLightGrayColor,
                            shape: RoundedRectangleBorder(
                                borderRadius: BorderRadius.circular(
                                    defaultRadius,
                                ),
                            ),
                        ),
                    ),
                ),
            ),
            child: Wrap(
                crossAxisAlignment:
WrapCrossAlignment.center,
                children: [
                    Icon(
                        Icons.male,
                        size: 20,
                        color: gender == 'male' ? kWhiteColor :
kGrayColor,
                ),
                const SizedBox(
                    width: 10,

```

```
        ),
        Text(
            'Laki-laki',
            style: whiteTextStyle.copyWith(
                color:
                    gender == 'male' ? kWhiteColor :
kGrayColor,
                ),
                ),
                ],
                ),
                ),
                ),
                ),
                ),
                ),
                ),
                const SizedBox(width: 16),
                Expanded(
                    child: SizedBox(
                        height: 46,
                        child: ElevatedButton(
                            onPressed: () {
                                setState(() {
                                    gender = 'female';
                                });
                            },
                            style: ElevatedButton.styleFrom(
                                elevation: 0,
                                primary:
                                    gender == 'female' ? kRedColor :
kLightGrayColor,
                                shape: RoundedRectangleBorder(
                                    borderRadius: BorderRadius.circular(
                                        defaultRadius,
                                    ),
                                    ),
                                    ),
                                    ),
                                    child: Wrap(
                                        mainAxisAlignment:
WrapCrossAlignment.center,
                                        children: [
                                            Icon(
                                                Icons.female,
                                                size: 20,
                                                color:
                                                    gender == 'female' ? kWhiteColor :
kGrayColor,
```

```
        ),
        const SizedBox(
            width: 10,
        ),
        Text(
            'Perempuan',
            style: whiteTextStyle.copyWith(
                color:
                    gender == 'female' ? kWhiteColor :
kGrayColor,
                ),
                ),
                ],
                ),
                ),
                ),
                ),
                ],
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Konsumsi Rokok per Hari',
                    controller: cigarettesController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Denyut Jantung',
                    controller: heartRateController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Gula Darah',
                    controller: glucoseController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Diabetes',
                    controller: diabetesController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Kolesterol',
                    controller: cholesterolController,
                ),
                const SizedBox(height: 16),
```

```

        CustomTextFormField(
            label: 'Sistolik',
            controller: systolicController,
        ),
        const SizedBox(height: 16),
        CustomTextFormField(
            label: 'Diastolik',
            controller: diastolicController,
        ),
        const SizedBox(height: 16),
        BlocConsumer<HeartDiseaseBloc, HeartDiseaseState>(
            listener: (context, state) {
                if (state is HeartDiseaseSuccess) {
                    Navigator.push(
                        context,
                        MaterialPageRoute(
                            builder: (context) => HeartDiseaseResultPage(
                                data: HeartDiseaseFormModel(
                                    age: int.parse(ageController.text),
                                    cigs: int.parse(cigarettesController.text),
                                    hRate: int.parse(heartRateController.text),
                                    gluc: int.parse(glucoseController.text),
                                    dia: int.parse(diabetesController.text),
                                    chol: int.parse(cholesterolController.text),
                                    sBP: int.parse(systolicController.text),
                                    dBP: int.parse(diastolicController.text),
                                    sex: gender == 'male' ? 1 : 0,
                                ),
                            ),
                        ),
                    );
                } else if (state is HeartDiseaseFailed) {
                    customShowSnackBar(context, state.e);
                }
            },
            builder: (context, state) {
                if (state is HeartDiseaseLoading) {
                    return CustomButton(
                        onPressed: () {},
                        child: SizedBox(
                            height: 21,
                            width: 21,
                        ),
                    );
                }
            },
        ),
    ),

```

```
        child: CircularProgressIndicator(
            color: kWhiteColor,
        ),
    ),
);
}
return CustomButton(
    onPressed: () {
        if (validate()) {
            context.read<HeartDiseaseBloc>().add(
                HeartDiseasePrediction(
                    HeartDiseaseFormModel(
                        age: int.parse(ageController.text),
                        cigs:
int.parse(cigarettesController.text),
                        hRate:
int.parse(heartRateController.text),
                        gluc:
int.parse(glucoseController.text),
                        dia:
int.parse(diabetesController.text),
                        chol:
int.parse(cholesterolController.text),
                        sBP:
int.parse(systolicController.text),
                        dBP:
int.parse(diastolicController.text),
                        sex: gender == 'male' ? 1 : 0,
                    ),
                ),
            );
        } else {
            customShowSnackBar(context, 'Semua field harus
diisi.');
        }
    },
    child: Text(
        'Cek',
        style: whiteTextStyle.copyWith(
            fontWeight: bold,
        ),
    ),
);
},
),
),
),
);
```

```

        ],
    ),
);
}

return Scaffold(
  backgroundColor: kWhiteColor,
  body: ListView(
    children: [
      appBar(),
      body(),
    ],
),
);
}
}

```

12. heart_disease_result_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/models/heart_disease_form_model.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';

import '../blocs/heart_disease/heart_disease_bloc.dart';

class HeartDiseaseResultPage extends StatelessWidget {
  const HeartDiseaseResultPage({required this.data, Key? key})
      : super(key: key);

  final HeartDiseaseFormModel data;

  @override
  Widget build(BuildContext context) {
    Widget appBar() {
      return const CustomAppBar(title: 'Hasil Prediksi');
    }

    Widget body() {
      return BlocBuilder<HeartDiseaseBloc, HeartDiseaseState>(
        builder: (context, state) {

```

```
if (state is HeartDiseaseSuccess) {
    return Container(
        margin: const EdgeInsets.only(
            left: 24,
            right: 24,
            bottom: 24,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                SizedBox(
                    width: double.infinity,
                    child: Column(
                        children: [
                            Image.asset(
                                'assets/health-checkup-4034059-
3337536.png',
                                height: 100,
                            ),
                            const SizedBox(height: 24),
                            Text(
                                'Probabilitas',
                                style: blackTextStyle.copyWith(
                                    fontSize: 16,
                                    fontWeight: bold,
                                ),
                            ),
                            const SizedBox(height: 10),
                            Wrap(
                                crossAxisAlignment:
WrapCrossAxisAlignment.center,
                                children: [
                                    Text(
                                        'Tidak memiliki: ',
                                        style: blackTextStyle,
                                    ),
                                    Text(
                                        state.data['probability'][0][0]
                                            .toStringAsFixed(3),
                                        style: blackTextStyle.copyWith(
                                            fontWeight: bold,
                                        ),
                                    ),
                                ],
                            ),
                        ],
                    ),
                ),
            ],
        ),
    );
}
```

```
        Wrap(
          mainAxisAlignment: MainAxisAlignment.end,
          children: [
            Text(
              'Memiliki: ',
              style: blackTextStyle,
            ),
            Text(
              state.data['probability'][0][1]
                .toStringAsFixed(3),
              style: blackTextStyle.copyWith(
                fontWeight: bold,
              ),
            ),
            ],
          ],
        ),
      ),
    ),
  ),
),
Container(
  margin: EdgeInsets.only(top: defaultMargin),
  child: Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
      Text(
        'Data Masukan',
        style: blackTextStyle.copyWith(
          fontSize: 16,
          fontWeight: bold,
        ),
      ),
      const SizedBox(height: 10),
      Wrap(
        mainAxisAlignment: MainAxisAlignment.end,
        children: [
          Text(
            'Umur: ',
            style: blackTextStyle,
          ),
          Text(
            data.age.toString(),
            style: blackTextStyle.copyWith(
              fontWeight: bold,
            ),
          ),
        ],
      ),
    ],
  ),
)
```

```
        ),
        ),
        ],
        ),
        Wrap(
            crossAxisAlignment:
WrapCrossAxisAlignment.center,
            children: [
                Text(
                    'Konsumsi Rokor per Hari: ',
                    style: blackTextStyle,
                ),
                Text(
                    data.cigs.toString(),
                    style: blackTextStyle.copyWith(
                        fontWeight: bold,
                    ),
                ),
                ],
            ),
            Wrap(
                crossAxisAlignment:
WrapCrossAxisAlignment.center,
                children: [
                    Text(
                        'Jenis Kelamin: ',
                        style: blackTextStyle,
                    ),
                    Text(
                        data.sex.toString(),
                        style: blackTextStyle.copyWith(
                            fontWeight: bold,
                        ),
                    ),
                ],
            ),
            Wrap(
                crossAxisAlignment:
WrapCrossAxisAlignment.center,
                children: [
                    Text(
                        'Denyut Jantung: ',
                        style: blackTextStyle,
                    ),
                    Text(

```

```
        data.hRate.toString(),
        style: blackTextStyle.copyWith(
            fontWeight: bold,
        ),
    ),
),
],
),
Wrap(
    crossAxisAlignment:
WrapCrossAlignment.center,
    children: [
        Text(
            'Gula Darah: ',
            style: blackTextStyle,
        ),
        Text(
            data.gluc.toString(),
            style: blackTextStyle.copyWith(
                fontWeight: bold,
            ),
        ),
        ],
),
),
Wrap(
    crossAxisAlignment:
WrapCrossAlignment.center,
    children: [
        Text(
            'Diabetes: ',
            style: blackTextStyle,
        ),
        Text(
            data.dia.toString(),
            style: blackTextStyle.copyWith(
                fontWeight: bold,
            ),
        ),
        ],
),
),
Wrap(
    crossAxisAlignment:
WrapCrossAlignment.center,
    children: [
        Text(
            'Kolesterol: ',

```

```
        style: blackTextStyle,
    ),
    Text(
        data.chol.toString(),
        style: blackTextStyle.copyWith(
            fontWeight: bold,
        ),
    ),
),
],
),
Wrap(
    crossAxisAlignment:
WrapCrossAxisAlignment.center,
    children: [
        Text(
            'Sistolik: ',
            style: blackTextStyle,
        ),
        Text(
            data.sBP.toString(),
            style: blackTextStyle.copyWith(
                fontWeight: bold,
            ),
        ),
    ],
),
Wrap(
    crossAxisAlignment:
WrapCrossAxisAlignment.center,
    children: [
        Text(
            'Diastolik: ',
            style: blackTextStyle,
        ),
        Text(
            data.dBP.toString(),
            style: blackTextStyle.copyWith(
                fontWeight: bold,
            ),
        ),
    ],
),
],
),
),
```

```

        ],
    ),
);
}
return Container();
},
);
}
}

return Scaffold(
backgroundColor: kWhiteColor,
body: ListView(
children: [
appBar(),
body(),
],
),
);
}
}
}

```

13. heart_rate_high_page.dart

```

import 'package:flutter/material.dart';

import '../shared/theme.dart';

class HeartRateHighPage extends StatelessWidget {
const HeartRateHighPage({required this.heartRate, Key? key})
: super(key: key);

final int heartRate;

@Override
Widget build(BuildContext context) {
Widget header() {
return Container(
margin: EdgeInsets.all(defaultMargin),
width: double.infinity,
child: Row(
children: [
GestureDetector(

```

```
        onTap: () {
            Navigator.pop(context);
        },
        child: Container(
            padding: const EdgeInsets.all(7),
            decoration: BoxDecoration(
                color: const Color(0xffffffff),
                borderRadius: BorderRadius.circular(12),
            ),
            child: const Icon(
                Icons.chevron_left,
                size: 21,
            ),
        ),
    ],
),
),
],
),
);
}
}

Widget content() {
    return Container(
        margin: EdgeInsets.only(
            left: defaultMargin,
            right: defaultMargin,
            bottom: defaultMargin,
        ),
        child: Column(
            children: [
                Text(
                    'Denyut Jantung Tinggi',
                    style: blackTextStyle.copyWith(
                        fontSize: 20,
                        fontWeight: semiBold,
                    ),
                ),
                const SizedBox(height: 10),
                Image.asset(
                    'assets/health-checkup-4034059-3337536.png',
                    height: 100,
                ),
                const SizedBox(height: 10),
                Text(
                    '$heartRate bpm',
                    style: blackTextStyle.copyWith(

```

```
        fontSize: 20,
        fontWeight: semiBold,
    ),
),
SizedBox(height: defaultMargin),
Column(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
        Text(
            'Denyut jantung kamu termasuk ke dalam kategori tinggi, atau biasa disebut dengan istilah takikardia. Takikardia terjadi jika denyut jantung melebihi 100 kali per menit atau bpm. Kondisi ini terjadi karena adanya gangguan listrik di jantung yang berperan dalam mengontrol irama denyut jantung.\nTakikardia bisa muncul tanpa menimbulkan komplikasi. Namun, jika tidak ditangani, denyut jantung yang tinggi bisa menyebabkan komplikasi serius, seperti stroke, gagal jantung, hentian jantung, dan bahkan kematian.\n',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
),
Text(
    'Penanganan Denyut Jantung Tinggi',
    style: blackTextStyle.copyWith(
        fontSize: 20,
        fontWeight: semiBold,
    ),
),
const SizedBox(height: 10),
Text(
    'Denyut jantung tinggi yang terjadi bukan karena penyakit, umumnya tidak membutuhkan pengobatan karena dapat membaik dengan sendirinya. Jika disebabkan oleh kondisi medis tertentu, penanganan denyut jantung tinggi akan disesuaikan dengan faktor penyebabnya.\nPenanganan yang dilakukan bertujuan untuk memperlambat denyut jantung tinggi hingga kembali dalam batas normal, mencegah denyut jantung tinggi kembali, menekan risiko komplikasi, dan mengobati penyakit mendasar yang dapat menyebabkan takikardia.\n\nPenanganan untuk takikardia meliputi:\n',
    style: blackTextStyle.copyWith(
        fontSize: 16,
    ),
),
Text(
```

```

        'Tindakan',
        style: blackTextStyle.copyWith(
            fontSize: 20,
            fontWeight: semiBold,
        ),
    ),
    const SizedBox(height: 10),
    Text(
        'Untuk meredakan denyut jantung tinggi, kamu bisa melakukan tindakan Manuver Vagal, yaitu melakukan gerakan seperti batuk, mengejan sebagaimana tengah buang air besar, dan menaruh es pada wajah untuk mempengaruhi saraf vagus yang mengatur denyut jantung.\n',
        style: blackTextStyle.copyWith(
            fontSize: 16,
        ),
    ),
    Text(
        'Obat-obatan',
        style: blackTextStyle.copyWith(
            fontSize: 20,
            fontWeight: semiBold,
        ),
    ),
    const SizedBox(height: 10),
    Text(
        'Untuk menormalkan denyut jantung tinggi atau cepat, dokter kemungkinan akan memberikan obat-obatan lain, seperti:\n',
        style: blackTextStyle.copyWith(
            fontSize: 16,
        ),
    ),
    Row(
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
            Text(
                ' • ',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
            ),
            Expanded(
                child: Text(
                    'Obat aniaritmia, untuk mengembalikan denyut

```

```
jantung normal. Obat ini diberikan jika Menouver Vagal tidak berhasil  
meredakan denyut jantung tinggi.',  
            style: blackTextStyle.copyWith(  
                fontSize: 16,  
            ),  
            ),  
            ),  
        ],  
    ),  
    Row(  
        mainAxisAlignment: CrossAxisAlignment.start,  
        children: [  
            Text(  
                ' • ',  
                style: blackTextStyle.copyWith(  
                    fontSize: 16,  
                ),  
            ),  
            Expanded(  
                child: Text(  
                    'Penghambat saluran kalsium dan penghambat  
beta, untuk mencegah denyut jantung tinggi terjadi lagi.',  
                    style: blackTextStyle.copyWith(  
                        fontSize: 16,  
                    ),  
                ),  
            ),  
            ],  
        ),  
        Row(  
            mainAxisAlignment: CrossAxisAlignment.start,  
            children: [  
                Text(  
                    ' • ',  
                    style: blackTextStyle.copyWith(  
                        fontSize: 16,  
                    ),  
                ),  
                Expanded(  
                    child: Text(  
                        'Obat pengencer darah, untuk membantu  
menurunkan risiko penggumpalan darah.\n',  
                        style: blackTextStyle.copyWith(  
                            fontSize: 16,  
                        ),  
                    ),  
                ),  
            ],  
        ),  
    ),  
);
```

```

        ),
        ),
        ],
        ),
        Text(
            'Menerapkan Gaya Hidup Sehat',
            style: blackTextStyle.copyWith(
                fontSize: 20,
                fontWeight: semiBold,
            ),
            ),
            const SizedBox(height: 10),
            Text(
                'Menerapkan gaya hidup sehat, seperti menjaga berat badan ideal, tetap aktif bergerak, mengurangi stress, dan menghindari rokok, merupakan beberapa cara agar jantung tetap sehat dan terhindar dari denyut jantung tinggi.\n\nJika denyut jantung tinggi terjadi secara berkelanjutan dan tidak diketahui dengan pasti apa penyebabnya, konsultasikan dengan dokter untuk mendapatkan pemeriksaan lebih lanjut.',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
                ),
                ),
                ],
                ),
                ],
                ),
                );
            }

            return Scaffold(
                body: ListView(
                    children: [
                        header(),
                        content(),
                    ],
                    ),
                    );
                );
            }
        }
    
```

14. heart_rate_information_page.dart

```
import 'package:flutter/material.dart';
import 'package:healio/widget/custom_app_bar.dart';

import '../shared/theme.dart';

class HeartRateInformationPage extends StatelessWidget {
  const HeartRateInformationPage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    Widget header() {
      return const CustomAppBar(
        title: 'Tentang Denyut Jantung',
      );
    }

    Widget columnItem(String value, bool isTitle) {
      return Column(
        children: [
          Container(
            padding: const EdgeInsets.all(10),
            child: Text(
              value,
              style: blackTextStyle.copyWith(
                fontSize: 16,
                fontWeight: isTitle ? bold : regular,
              ),
            ),
          ),
        ],
      );
    }

    Widget content() {
      return Container(
        margin: EdgeInsets.only(
          right: defaultMargin,
          left: defaultMargin,
        ),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            ClipRRect(

```

```

borderRadius: BorderRadius.circular(12),
child: Image.asset(
  'assets/hr_thumbnail.jpg',
),
),
),
SizedBox(
  height: defaultMargin,
),
Text(
  'Detak Jantung',
  style: blackTextStyle.copyWith(
    fontSize: 18,
    fontWeight: bold,
  ),
),
const SizedBox(
  height: 10,
),
Text(
  'Jantung merupakan organ vital pada tubuh manusia. Fungsi jantung adalah memompa darah ke seluruh tubuh, sehingga berbagai organ dan sistem tubuh Anda bisa bekerja sebagaimana mestinya.\n\nSelain tekanan darah, salah satu indikator penting dari kesehatan jantung adalah detak jantung. Detak jantung adalah berapa kali jantung Anda berdenyut dalam satu menit. Adapun detak jantung seseorang dipengaruhi oleh berbagai faktor, seperti usia, ukuran tubuh, kondisi jantung, cuaca atau temperatur udara, aktivitas fisik, emosi, dan obat-obatan tertentu.',
  style: blackTextStyle.copyWith(
    fontSize: 16,
  ),
),
SizedBox(
  height: defaultMargin,
),
Text(
  'Tabel Detak Jantung Berdasarkan Usia',
  style: blackTextStyle.copyWith(
    fontSize: 18,
    fontWeight: bold,
  ),
),
const SizedBox(
  height: 10,
),

```

```
        Center(
            child: Table(
                border: TableBorder.all(
                    color: Colors.black,
                    style: BorderStyle.solid,
                    width: 1,
                ),
                children: [
                    TableRow(
                        children: [
                            columnItem('Usia', true),
                            columnItem('Normal (bpm)', true),
                        ],
                    ),
                    TableRow(
                        children: [
                            columnItem('Bayi', false),
                            columnItem('70 - 130', false),
                        ],
                    ),
                    TableRow(
                        children: [
                            columnItem('Anak - anak', false),
                            columnItem('80 - 110', false),
                        ],
                    ),
                    TableRow(
                        children: [
                            columnItem('Dewasa', false),
                            columnItem('60 - 100', false),
                        ],
                    ),
                    ],
                ),
            ),
            SizedBox(
                height: defaultMargin,
            ),
            const Divider(),
        ],
    ),
);
}

Widget disclaimer() {
```

```

        return Container(
            margin: EdgeInsets.all(defaultMargin),
            child: Column(
                mainAxisAlignment: MainAxisAlignment.start,
                children: [
                    Row(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                            Icon(
                                Icons.warning,
                                size: 16,
                                color: kRedColor,
                            ),
                            const SizedBox(
                                width: 5,
                            ),
                            Text(
                                'Disclaimer',
                                style: blackTextStyle.copyWith(
                                    fontSize: 18,
                                    fontWeight: bold,
                                ),
                            ),
                            ],
                        ),
                    const SizedBox(
                        height: 10,
                    ),
                    Text(
                        'Harap diingat bahwa pemantauan detak jantung hanya
                        untuk kebugaran dan kesehatan, bukan untuk diagnosis atau perawatan
                        kondisi medis apapun. Jika Anda memiliki kekhawatiran tentang
                        jantung Anda, pastikan untuk berkonsultasi dengan profesional
                        medis.',
                        style: blackTextStyle.copyWith(
                            fontSize: 16,
                        ),
                    ),
                    ],
                );
            }
        );

        return Scaffold(
            body: ListView(

```

```

        children: [
            header(),
            content(),
            disclaimer(),
        ],
    ),
);
}
}

```

15. heart_rate_low_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/health/health_bloc.dart';
import 'package:health/health.dart';

import '../shared/theme.dart';

class HeartRateLowPage extends StatelessWidget {
    const HeartRateLowPage({required this.heartRate, Key? key}) : super(key: key);

    final int heartRate;

    @override
    Widget build(BuildContext context) {
        Widget header() {
            return Container(
                margin: EdgeInsets.all(defaultMargin),
                width: double.infinity,
                child: Row(
                    children: [
                        GestureDetector(
                            onTap: () {
                                Navigator.pop(context);
                            },
                            child: Container(
                                padding: const EdgeInsets.all(7),
                                decoration: BoxDecoration(
                                    color: const Color(0xffefefef),
                                    borderRadius: BorderRadius.circular(12),

```

```
        ),
        child: const Icon(
            Icons.chevron_left,
            size: 21,
        ),
    ),
),
],
),
);
}

Widget content() {
    return Container(
        margin: EdgeInsets.only(
            left: defaultMargin,
            right: defaultMargin,
            bottom: defaultMargin,
        ),
        child: Column(
            children: [
                Text(
                    'Denyut Jantung Rendah',
                    style: blackTextStyle.copyWith(
                        fontSize: 20,
                        fontWeight: semiBold,
                    ),
                ),
                const SizedBox(height: 10),
                Image.asset(
                    'assets/health-checkup-4034059-3337536.png',
                    height: 100,
                ),
                const SizedBox(height: 10),
                Text(
                    '$heartRate bpm',
                    style: blackTextStyle.copyWith(
                        fontSize: 20,
                        fontWeight: semiBold,
                    ),
                ),
                SizedBox(height: defaultMargin),
                Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    children: [

```

```

    Text(
        'Denyut jantung kamu termasuk ke dalam kategori rendah, atau biasa disebut dengan istilah bradikardia. Bradikardia adalah kondisi ketika jantung berdetak lebih lambat dari biasanya.\n\nMelambatnya denyut jantung umumnya merupakan hal yang normal. Kondisi tersebut dapat terjadi pada orang yang sedang tidur, remaja, atau atlet. Namun, jika disertai dengan gejala pusing atau sesak nafas, denyut jantung yang melambat bisa menjadi tandanya gangguan pada aktivitas listrik jantung.\n\n',
        style: blackTextStyle.copyWith(
            fontSize: 16,
        ),
    ),
    Text(
        'Pencegahan Bradikardia',
        style: blackTextStyle.copyWith(
            fontSize: 20,
            fontWeight: semiBold,
        ),
    ),
    const SizedBox(height: 10),
    Text(
        'Bradikardia dapat dicegah dengan menghindari faktor-faktor yang dapat meningkatkan risiko terjadinya kondisi ini. Caranya adalah dengan mengubah gaya hidup agar lebih sehat, yaitu dengan melakukan langkah sederhana berikut ini:\n',
        style: blackTextStyle.copyWith(
            fontSize: 16,
        ),
    ),
    Row(
        mainAxisAlignment: CrossAxisAlignment.start,
        children: [
            Text(
                '• ',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
            ),
            Expanded(
                child: Text(
                    'Menghindari kebiasaan merokok.',
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                    ),
                ),
            ),
        ],
    ),

```

```
        ),
        ),
    ],
),
Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
        Text(
            '• • ',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
        Expanded(
            child: Text(
                'Menghindari penggunaan NAPZA.',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
            ),
        ),
        ],
),
Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
        Text(
            '• • ',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
        Expanded(
            child: Text(
                'Membatasi konsumsi alkohol.',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
            ),
        ),
        ],
),
Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
```

```
        Text(
          ' • • ',
          style: blackTextStyle.copyWith(
            fontSize: 16,
          ),
        ),
      ),
      Expanded(
        child: Text(
          'Menghindari stres.',
          style: blackTextStyle.copyWith(
            fontSize: 16,
          ),
        ),
      ),
    ],
),
Row(
  mainAxisAlignment: CrossAxisAlignment.start,
  children: [
    Text(
      ' • • ',
      style: blackTextStyle.copyWith(
        fontSize: 16,
      ),
    ),
    Expanded(
      child: Text(
        'Menjaga berat badan ideal.',
        style: blackTextStyle.copyWith(
          fontSize: 16,
        ),
      ),
    ),
  ],
),
Row(
  mainAxisAlignment: CrossAxisAlignment.start,
  children: [
    Text(
      ' • • ',
      style: blackTextStyle.copyWith(
        fontSize: 16,
      ),
    ),
  ],
),
Expanded(
```

```

        child: Text(
            'Berolahraga secara rutin.',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
    ),
),
],
),
Row(
    crossAxisAlignment: CrossAxisAlignment.start,
    children: [
        Text(
            '• • •',
            style: blackTextStyle.copyWith(
                fontSize: 16,
            ),
        ),
        Expanded(
            child: Text(
                'Mengonsumsi makanan bergizi seimbang dan
rendah garam.',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                ),
            ),
        ),
    ],
),
],
),
],
),
),
),
);
}

return Scaffold(
    body: ListView(
        children: [
            header(),
            content(),
        ],
    ),
);
}

```

```
}
```

16. heart_rate_page.dart

```
import 'dart:developer';
import 'dart:math' as math;

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/models/health_model.dart';
import 'package:healio/services/health_service.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_app_bar.dart';
import 'package:healio/widget/health_data_item.dart';
import 'package:healio/widget/more_button.dart';
import 'package:health/health.dart';
import 'package:intl/intl.dart';

import '../blocs/auth/auth_bloc.dart';
import '../blocs/health/health_bloc.dart';
import '../services/push_notification_service.dart';
import '../shared/shared_methods.dart';

class HeartRatePage extends StatefulWidget {
    const HeartRatePage({Key? key}) : super(key: key);

    @override
    State<HeartRatePage> createState() => _HeartRatePageState();
}

late HealthModel hrFromFirebase;

List<HealthDataPoint> dataHeartRate = [];

int age = 0;

String category = '';

class _HeartRatePageState extends State<HeartRatePage> {
    @override
    void initState() {
        super.initState();
```

```

        getDataFromFirebase();
        fetchData();
        getAge();
    }

    void fetchData() {
        context.read<HealthBloc>().add(HealthGetHeartRate());
        final healthState = context.read<HealthBloc>().state;
        if (healthState is HealthHeartRateSuccess) {
            dataHeartRate = healthState.heartRate;
        }
    }

    void getDataFromFirebase() async {
        hrFromFirebase = await
    HealthService().getHeartRateFromFirebase();
    }

    void getAge() {
        final authState = context.read<AuthBloc>().state;
        if (authState is AuthSuccess) {
            age =
        calculateAge(DateTime.parse(authState.user.dateOfBirth));
        }
    }

    void updateHeartRate() async {
        if (hrFromFirebase.value ==
            getValueFromHealthValue(dataHeartRate.first.value) &&
            hrFromFirebase.dateFrom ==
        dataHeartRate.first.dateFrom.toString()) {
            log('Sama');
            log(age.toString());
        } else {
            log('beda');
            await HealthService().updateHeartRateFromFirebase(
                getValueFromHealthValue(dataHeartRate.first.value),
                dataHeartRate.first.dateFrom.toString());
            category =
        setCategory(getValueFromHealthValue(dataHeartRate.first.value),
            'heart_rate', age);
            log(category);
            if (category == 'rendah') {
                customShowSnackBar(context, 'Denyut jantung kamu rendah');
                await PushNotificationService().pushNotification(

```

```

        'Denyut      Jantung
${getValueFromHealthValue(dataHeartRate.first.value)}',
    'Denyut jantung kamu dibawah nilai normal nih :(.',
);
} else if (category == 'tinggi') {
    customShowSnackBar(context, 'Denyut jantung kamu tinggi');
    await PushNotificationService().pushNotification(
        'Denyut      Jantung
${getValueFromHealthValue(dataHeartRate.first.value)}',
    'Denyut jantung kamu tinggi nih :(.',
);
}
}

syncData() async {
    fetchData();
    getDataFromFirebase();
    updateHeartRate();
}

@Override
Widget build(BuildContext context) {
    String formattedDate = DateFormat("EEEE, d
MMMM").format(DateTime.now());

    Widget header() {
        return CustomAppBar(
            title: 'Denyut Jantung',
            action: GestureDetector(
                onTap: () {
                    Navigator.pop(context);
                },
                child: GestureDetector(
                    onTap: () {
                        syncData();
                    },
                    child: Container(
                        padding: const EdgeInsets.all(10),
                        decoration: BoxDecoration(
                            color: kLightGrayColor,
                            borderRadius: BorderRadius.circular(
                                defaultRadius,
                            ),
                    ),
                ),
            ),
        );
    }
}

```

```

        child: Image.asset(
            'assets/fi_refresh-cw.png',
            height: 20,
        ),
    ),
),
),
),
),
);
}

Widget report() {
    return BlocBuilder<HealthBloc, HealthState>(
        builder: (context, state) {
            if (state is HealthHeartRateSuccess) {
                List<int> listValue = state.heartRate
                    .map((e) => getValueFromHealthValue(e.value))
                    .toList();
                return Container(
                    margin: EdgeInsets.symmetric(
                        horizontal: defaultMargin,
                    ),
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.start,
                        children: [
                            Text(
                                'Ringkasan',
                                style: blackTextStyle.copyWith(
                                    fontSize: 16,
                                    fontWeight: bold,
                                ),
                            ),
                            const SizedBox(height: 10),
                            Container(
                                padding: const EdgeInsets.all(16),
                                decoration: BoxDecoration(
                                    color: kLightGrayColor,
                                    borderRadius: BorderRadius.circular(
                                        defaultRadius,
                                    ),
                            ),
                            ),
                            child: Column(
                                mainAxisAlignment: MainAxisAlignment.start,
                                children: [
                                    Text(
                                        formattedDate,

```

```
        style: grayTextStyle,
    ),
    const SizedBox(height: 24),
    Column(
        children: [
            Container(
                height: 56,
                width: 56,
                decoration: BoxDecoration(
                    color: kWhiteColor,
                    shape: BoxShape.circle,
                ),
                padding: const EdgeInsets.all(16),
                child: Image.asset(
                    'assets/fi_heart.png',
                    color: kRedColor,
                ),
            ),
            const SizedBox(height: 24),
            Row(
                children: [
                    Expanded(
                        child: Container(
                            decoration: BoxDecoration(
                                border: Border(
                                    right: BorderSide(
                                        width: 1,
                                        color: kGrayColor.withOpacity(0.3),
                                    ),
                                ),
                            ),
                        ),
                    ),
                    child: Column(
                        crossAxisAlignment: CrossAxisAlignment.center,
                        children: [
                            Text(
                                'Minimum:',
                                style: grayTextStyle.copyWith(
                                    fontSize: 12,
                                ),
                            ),
                            Wrap(
                                crossAxisAlignment:
```

```
WrapCrossAlignment.end,
            children: [
              Text(
                '${state.heartRate.isEmpty ? 0 : listValue.reduce(math.min)}',
                // '',
                style: blackTextStyle.copyWith(
                  fontSize: 20,
                  fontWeight: bold,
                ),
              ),
              Text(
                'BPM',
                style: grayTextStyle.copyWith(),
              ),
            ],
          ),
        ],
      ),
    ),
  ),
),
Expanded(
  child: Container(
    decoration: BoxDecoration(
      border: Border(
        right: BorderSide(
          width: 1,
        ),
        color: kGrayColor.withOpacity(0.3),
      ),
    ),
  ),
),
child: Column(
  crossAxisAlignment: CrossAxisAlignmentAlignment.center,
  children: [
    Text(
      'Terbaru:',
      style: grayTextStyle.copyWith(
        fontSize: 12,
      ),
    ),
  ],
),
```

```
        ),
      Wrap(
        mainAxisAlignment:
          WrapCrossAlignment.end,
        children: [
          Text(
            '${state.heartRate.isEmpty ? '' : getValueFromHealthValue(state.heartRate.first.value)}',
            style: blackTextStyle.copyWith(
              fontSize: 20,
              fontWeight: bold,
            ),
          ),
          Text(
            'BPM',
            style: grayTextStyle.copyWith(),
          ),
          ],
        ),
        ),
        ],
      ),
      ),
      ),
      ),
      ),
      ),
      ),
      ),
      ),
      ),
      Expanded(
        child: Column(
          mainAxisAlignment:
            MainAxisAlignment.center,
          children: [
            Text(
              'Maksimum:',
              style: grayTextStyle.copyWith(
                fontSize: 12,
              ),
            ),
            ],
          ),
        ),
      ),
      Wrap(
        mainAxisAlignment:
          WrapCrossAlignment.end,
        children: [
          Text(

```

```
'${state.heartRate.isEmpty ? 0 : listValue.reduce(math.max)}',
                // '',
                style:
blackTextStyle.copyWith(
                    fontSize: 20,
                    fontWeight: bold,
                    ),
                ),
Text(
    'BPM',
    style:
grayTextStyle.copyWith(),
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    ),
                    ],
                    );
}
return Container();
},
);
}
}

Widget more() {
return const MoreButton(
text: 'Tentang Denyut Jantung',
route: '/heart-rate-information',
);
}

Widget history() {
return BlocBuilder<HealthBloc, HealthState>(
builder: (context, state) {
```

```

        if (state is HealthHeartRateSuccess) {
            return Container(
                margin: EdgeInsets.all(
                    defaultMargin,
                ),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.start,
                    children: [
                        Row(
                            children: [
                                Text(
                                    'Data',
                                    style: blackTextStyle.copyWith(
                                        fontSize: 16,
                                        fontWeight: bold,
                                    ),
                                ),
                                const Spacer(),
                                GestureDetector(
                                    onTap: () {
                                        Navigator.pushNamed(context, '/all-heart-
rate');
                                    },
                                ),
                                Text(
                                    'Semua',
                                    style: grayTextStyle.copyWith(
                                        fontSize: 14,
                                        fontWeight: bold,
                                    ),
                                ),
                            ],
                        ),
                        state.heartRate.isEmpty
                            ? Center(
                                child: Container(
                                    margin: const EdgeInsets.only(
                                        top: 56,
                                    ),
                                ),
                                padding: const
                            EdgeInsets.symmetric(horizontal: 36),
                                child: Column(
                                    children: [
                                        Image.asset(
                                            'assets/document.png',
                                           

```

```

        height: 100,
    ),
    const SizedBox(height: 10),
    Text(
        'Data Kosong',
        style: blackTextStyle.copyWith(
            fontSize: 20,
            fontWeight: bold,
        ),
    ),
    const SizedBox(height: 5),
    Text(
        'Data denyut jantung kamu hari ini
masih kosong nih, yuk lakukan pengecekan pada smartwatch kamu.',
        style: grayTextStyle.copyWith(),
        textAlign: TextAlign.center,
    ),
],
),
),
),
)
),
:
Column(
    children: state.heartRate
        .map((health) =>
HealthDataItem(health))
        .toList(),
),
],
),
);
}
return Container();
},
);
}
}

return Scaffold(
body: ListView(
children: [
header(),
report(),
more(),
history(),
],
),

```

```
    );
}
}
```

17. home_page.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:heilio/models/article_model.dart';
import 'package:heilio/pages/blood_oxygen_low_handling_page.dart';
import 'package:heilio/pages/heart_rate_high_page.dart';
import 'package:heilio/pages/heart_rate_low_page.dart';
import 'package:heilio/shared/shared_methods.dart';
import 'package:heilio/shared/theme.dart';
import 'package:heilio/widget/article_card.dart';

import '../blocs/article/article_bloc.dart';
import '../blocs/auth/auth_bloc.dart';
import '../blocs/health/health_bloc.dart';

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

int age = 0;

class _HomePageState extends State<HomePage> {
  @override
  void initState() {
    super.initState();
    context.read<HealthBloc>().add(HealthGetLandingPage());
    getAge();
  }

  void getAge() {
    final authState = context.read<AuthBloc>().state;
    if (authState is AuthSuccess) {
      age = calculateAge(DateTime.parse(authState.user.dateOfBirth));
    }
  }
}
```

```
        }
    }

    @override
    Widget build(BuildContext context) {
        Widget header() {
            return BlocBuilder<AuthBloc, AuthState>(
                builder: (context, state) {
                    if (state is AuthSuccess) {
                        age = calculateAge(DateTime.parse(state.user.dateOfBirth));
                        return Container(
                            margin: EdgeInsets.all(
                                defaultMargin,
                            ),
                            child: Row(
                                mainAxisAlignment: MainAxisAlignment.start,
                                children: [
                                    Column(
                                        mainAxisAlignment: MainAxisAlignment.start,
                                        children: [
                                            Text(
                                                'Halo,\n${state.user.name}',
                                                style: blackTextStyle.copyWith(
                                                    fontSize: 26,
                                                    fontWeight: bold,
                                                ),
                                            ),
                                            const SizedBox(height: 10),
                                            Text(
                                                'Bagaimana kesehatanmu hari ini?',
                                                style: grayTextStyle.copyWith(),
                                            ),
                                        ],
                                    ),
                                    const Spacer(),
                                    Container(
                                        height: 64,
                                        width: 64,
                                        decoration: const BoxDecoration(
                                            image: DecorationImage(
                                                image: AssetImage(
                                                    'assets/avatar.png',
                                                ),
                                            ),
                                        ),
                                    ),
                                ],
                            ),
                        );
                    }
                }
            );
        }
    }
}
```

```

                shape: BoxShape.circle,
            ),
        ),
    ],
),
);
}
return Container();
},
);
}
}

Widget today() {
return BlocBuilder<HealthBloc, HealthState>(
builder: (context, state) {
if (state is HealthLandingPageSuccess) {
return Container(
margin: EdgeInsets.symmetric(
horizontal: defaultMargin,
),
child: Column(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Text(
'Hari Ini',
style: blackTextStyle.copyWith(
fontSize: 16,
fontWeight: bold,
),
),
const SizedBox(height: 10),
Row(
crossAxisAlignment: CrossAxisAlignment.start,
children: [
Expanded(
child: Container(
padding: const EdgeInsets.all(16),
decoration: BoxDecoration(
color: kLightGrayColor,
borderRadius: BorderRadius.circular(
12,
),
),
child: Column(
crossAxisAlignment:

```

```

CrossAxisAlignment.start,
    children: [
        Container(
            height: 42,
            width: 42,
            decoration: BoxDecoration(
                color: kWhiteColor,
                shape: BoxShape.circle,
            ),
            padding: const EdgeInsets.all(12),
            margin: const EdgeInsets.only(
                bottom: 10,
            ),
            child: Image.asset(
                'assets/fi_heart.png',
                color: kRedColor,
            ),
        ),
        Text(
            'Denyut Jantung',
            style: grayTextStyle.copyWith(
                fontSize: 14,
            ),
        ),
        const SizedBox(height: 5),
        Wrap(
            crossAxisAlignment:
WrapCrossAxisAlignment.end,
            children: [
                Text(
                    // '$newestHeartRate ',
                    '${state.heartRate.isEmpty ? 0 : getValueFromHealthValue(state.heartRate.first.value)}',
                    style: blackTextStyle.copyWith(
                        fontSize: 20,
                        fontWeight: bold,
                    ),
                ),
                Text(
                    'BPM',
                    style: grayTextStyle.copyWith(
                        fontWeight: bold),
                ),
            ],
        ),
    ],
),

```

```
        ],
      ),
    ),
  ),
),
const SizedBox(width: 16),
Expanded(
  child: Container(
    padding: const EdgeInsets.all(16),
    decoration: BoxDecoration(
      color: kLightGrayColor,
      borderRadius: BorderRadius.circular(
        defaultRadius,
      ),
    ),
  ),
  child: Column(
    crossAxisAlignment:
CrossAxisAlignment.start,
    children: [
      Container(
        height: 42,
        width: 42,
        decoration: BoxDecoration(
          color: kWhiteColor,
          shape: BoxShape.circle,
        ),
        padding: const EdgeInsets.all(12),
        margin: const EdgeInsets.only(
          bottom: 10,
        ),
        child: Image.asset(
          'assets/fi_droplet.png',
          color: kPrimaryColor,
        ),
      ),
      Text(
        'Saturasi Oksigen',
        style: grayTextStyle.copyWith(
          fontSize: 14,
        ),
      ),
      const SizedBox(height: 5),
      Wrap(
        crossAxisAlignment:
WrapCrossAxisAlignment.center,
        children: [
```

```

        Text(
            '${state.bloodOxygen.isEmpty ? 0
: getValueFromHealthValue(state.bloodOxygen.first.value)}',
            style: blackTextStyle.copyWith(
                fontSize: 20,
                fontWeight: bold,
            ),
        ),
        Text(
            '%',
            style: grayTextStyle.copyWith(
                fontSize: 20,
            ),
        ),
        ],
        ),
        ],
        ],
        ],
        ],
        ],
        ],
        ],
        ],
        ],
        );
    }
    return Container();
},
);
}
}

Widget announce() {
return BlocBuilder<HealthBloc, HealthState>(
builder: (context, state) {
if (state is HealthLandingPageSuccess) {
String hrCategory = state.heartRate.isNotEmpty
? setCategory(
getValueFromHealthValue(state.heartRate.first.value),
'heart_rate',
age,
)
: 'empty';
String boCategory = state.bloodOxygen.isNotEmpty
? setCategory(

```

```

getValueFromHealthValue(state.bloodOxygen.first.value),
    'blood_oxygen',
    age,
)
: 'empty';
int hrValue =
getValueFromHealthValue(state.heartRate.first.value);
int boValue =

getValueFromHealthValue(state.bloodOxygen.first.value);
return Container(
    margin: EdgeInsets.all(defaultMargin),
    child: Column(
        mainAxisAlignment: CrossAxisAlignment.start,
        children: [
            Text(
                'Notifikasi',
                style: blackTextStyle.copyWith(
                    fontSize: 16,
                    fontWeight: bold,
                ),
            ),
            const SizedBox(height: 10),
            Container(
                width: double.infinity,
                decoration: BoxDecoration(
                    color: kRedColor,
                    borderRadius: BorderRadius.circular(
                        defaultRadius,
                    ),
                ),
                padding: const EdgeInsets.all(16),
                child: Row(
                    children: [
                        Image.asset(
                            'assets/heart-rate-monitor.png',
                            height: 75,
                        ),
                        const SizedBox(width: 10),
                        Expanded(
                            child: Column(
                                mainAxisAlignment:
CrossAxisAlignment.start,
                                children: [

```

```

        Text(
          hrCategory == 'tinggi'
            ? 'Denyut Jantung Tinggi'
            : hrCategory == 'rendah'
            ? 'Denyut Jantung Rendah'
            : hrCategory == 'normal'
            ? 'Denyut Jantung
Normal'
            : 'Data Denyut Jantung
Kosong',
          style: whiteTextStyle.copyWith(
            fontSize: 16,
            fontWeight: bold,
          ),
        ),
        const SizedBox(height: 10),
        Text(
          hrCategory == 'tinggi'
            ? 'Wah denyut jantung kamu
tinggi 😊 Yuk baca tips berikut sebagai penanganan pertama pada
kondisi kamu.'
            : hrCategory == 'rendah'
            ? 'Wah denyut jantung kamu
rendah 😊 Yuk baca tips berikut sebagai penanganan pertama pada
kondisi kamu.'
            : hrCategory == 'normal'
            ? 'Denyut Jantung kamu
normal nih, tetap pertahankan ya.'
            : 'Data Denyut jantung
masih kosong nih, yuk lakukan pengecekan pada smartwatch kamu.',
          style: whiteTextStyle,
        ),
        const SizedBox(height: 10),
        hrCategory == 'rendah' || hrCategory
== 'tinggi'
        ? Row(
          mainAxisAlignment:
MainAxisAlignment.end,
          children: [
            Container(
              decoration: BoxDecoration(
                color: kWhiteColor,
                borderRadius:
BorderRadius.circular(
              defaultRadius,

```

```
        ),
        ),
        padding: const
EdgeInsets.symmetric(
    horizontal: 16,
    vertical: 10,
),
child: GestureDetector(
onTap: () {
    Navigator.push(
        context,
        MaterialPageRoute(
            builder: (content)
=>
        hrCategory ==
'rendah'
?
HeartRateLowPage(
heartRate:
hrValue,
)
:
HeartRateHighPage(
heartRate:
hrValue,
),
),
);
},
child: Text(
'Lihat',
style:
blackTextStyle.copyWith(
fontWeight: bold,
),
),
),
),
],
)
: const SizedBox(),
```

```

        ],
        ),
        ),
        ],
        ),
        ),
        ),
        const SizedBox(height: 10),
Container(
    width: double.infinity,
    decoration: BoxDecoration(
        color: kPrimaryColor,
        borderRadius: BorderRadius.circular(
            defaultRadius,
        ),
    ),
    padding: const EdgeInsets.all(16),
    child: Row(
        children: [
            Expanded(
                child: Column(
                    crossAxisAlignment:
CrossAxisAlignment.start,
                    children: [
                        Text(
                            boCategory == 'rendah'
                                ? 'Saturasi Oksigen Rendah'
                                : boCategory == 'normal'
                                    ? 'Saturasi Oksigen Normal'
                                    : 'Data Saturasi Oksigen
Kosong',
                            style: whiteTextStyle.copyWith(
                                fontSize: 16,
                                fontWeight: bold,
                            ),
                        ),
                        const SizedBox(height: 10),
                        Text(
                            boCategory == 'rendah'
                                ? 'Wah Saturasi Oksigen kamu
rendah 😞 Yuk baca tips berikut sebagai penanganan pertama pada
kondisi kamu.'
                                : boCategory == 'normal'
                                    ? 'Saturasi Oksigen kamu
normal nih, tetap pertahankan ya.'
                                    : 'Data Saturasi Oksigen
'
                        )
                    ],
                ),
            ),
        ],
    ),
)

```

```

masih kosong nih, yuk lakukan pengecekan pada smartwatch kamu.',  

        style: whiteTextStyle,  

    ),  

    boCategory == 'rendah'  

    ? GestureDetector(  

        onTap: () {  

            Navigator.push(  

                context,  

                MaterialPageRoute(  

                    builder: (context) =>  

BloodOxygenLowHandllingPage(  

                bloodOxygen: boValue,  

                ),  

                ),  

                );  

            },  

            child: Container(  

                decoration: BoxDecoration(  

                    color: kWhiteColor,  

                    borderRadius:  

BorderRadius.circular(  

                defaultRadius,  

                ),  

                ),  

                margin: const  

EdgeInsets.only(top: 10),  

                padding: const  

EdgeInsets.symmetric(  

                horizontal: 16,  

                vertical: 10,  

                ),  

                child: Text(  

                    'Lihat',  

                    style:  

blackTextStyle.copyWith(  

                fontWeight: bold,  

                ),  

                ),  

                ),  

                )  

: const SizedBox(),  

                ],  

                ),  

                ),

```

```

        Image.asset(
            'assets/announce2_image.png',
            height: 75,
        ),
    ],
),
),
),
],
),
);
}
return Container();
},
);
}
}

Widget article() {
return Container(
margin: EdgeInsets.only(
    left: defaultMargin,
    right: defaultMargin,
    bottom: defaultMargin,
),
child: Column(
children: [
Row(
mainAxisAlignment: MainAxisAlignment.spaceBetween,
children: [
Text(
    'Artikel',
    style: blackTextStyle.copyWith(
        fontSize: 16,
        fontWeight: bold,
    ),
),
GestureDetector(
onTap: () {
    Navigator.pushNamed(context, '/article');
},
child: Text(
    'Semua',
    style: grayTextStyle.copyWith(
        fontWeight: bold,
    ),
),
),
]
),
]
),
)
}
}

```

```

        ),
    ],
),
Column(
    children: List.generate(
        5,
        (index) => ArticleCard(
            listArticle[index],
        ),
    ),
),
],
),
);
}
}

return Scaffold(
    backgroundColor: kWhiteColor,
    body: ListView(
        children: [
            header(),
            today(),
            announce(),
            article(),
        ],
    ),
);
}
}

```

18. main_page.dart

```

import 'package:flutter/material.dart';
import 'package:heilio/blocs/article/article_bloc.dart';
import 'package:heilio/blocs/health/health_bloc.dart';
import 'package:heilio/pages/health_page.dart';
import 'package:heilio/pages/home_page.dart';
import 'package:heilio/pages/user_page.dart';
import 'package:heilio/shared/theme.dart';
import 'package:flutter_bloc/flutter_bloc.dart';

import '../blocs/page/page_cubit.dart';

```

```
class MainPage extends StatefulWidget {
  const MainPage({Key? key}) : super(key: key);

  @override
  State<MainPage> createState() => _MainPageState();
}

class _MainPageState extends State<MainPage> {
  @override
  void initState() {
    super.initState();
    fetchData();
  }

  fetchData() {
    context.read<HealthBloc>().add(HealthGetLandingPage());
    context.read<HealthBloc>().add(HealthGetHeartRate());
    context.read<HealthBloc>().add(HealthGetBloodOxygen());
    context.read<ArticleBloc>().add(const ArticleGet('heart rate'));
  }

  @override
  Widget build(BuildContext context) {
    List pages = const [
      HomePage(),
      HealthPage(),
      UserPage(),
    ];

    return BlocBuilder<PageCubit, int>(
      builder: (context, currentIndex) {
        return RefreshIndicator(
          onRefresh: _pullToRefresh,
          child: Scaffold(
            body: pages[currentIndex],
            bottomNavigationBar: BottomNavigationBar(
              type: BottomNavigationBarType.fixed,
              backgroundColor: Colors.white,
              onTap: (value) {
                context.read<PageCubit>().setPage(value);
              },
              currentIndex: currentIndex,
              showSelectedLabels: false,
              showUnselectedLabels: false,
            ),
          ),
        );
      },
    );
  }
}
```

```
        items: [
          BottomNavigationBarItem(
            label: 'Home',
            icon: Column(
              children: [
                Image.asset(
                  currentIndex == 0
                    ? 'assets/icon_home_active.png'
                    : 'assets/icon_home.png',
                  color: currentIndex == 0 ? kPrimaryColor :
kGrayColor,
                  width: 20,
                  height: 20,
                ),
                const SizedBox(
                  height: 5,
                ),
                Text(
                  'Home',
                  style: blackTextStyle.copyWith(
                    fontSize: 12,
                    fontWeight: currentIndex == 0 ? bold :
reguler,
                    color: currentIndex == 0 ? kPrimaryColor :
kGrayColor,
                  ),
                ),
              ],
            ),
          ),
          BottomNavigationBarItem(
            label: 'Kesehatan',
            icon: Column(
              children: [
                Image.asset(
                  currentIndex == 1
                    ? 'assets/icon_heart_active.png'
                    : 'assets/icon_heart.png',
                  color: currentIndex == 1 ? kPrimaryColor :
kGrayColor,
                  height: 20,
                  width: 20,
                ),
                const SizedBox(
                  height: 5,
```

```
        ),
        Text(
          'Kesehatan',
          style: blackTextStyle.copyWith(
            fontSize: 12,
            fontWeight: currentIndex == 1 ? bold :
reguler,
            color: currentIndex == 1 ? kPrimaryColor :
kGrayColor,
          ),
        ),
      ],
    ),
  ),
),
BottomNavigationBarItem(
  label: 'User',
  icon: Column(
    children: [
      Image.asset(
        currentIndex == 2
        ? 'assets/icon_user_active.png'
        : 'assets/icon_user.png',
        color: currentIndex == 2 ? kPrimaryColor :
kGrayColor,
        width: 20,
        height: 20,
      ),
      const SizedBox(
        height: 5,
      ),
      Text(
        'Akun',
        style: blackTextStyle.copyWith(
          fontSize: 12,
          fontWeight: currentIndex == 2 ? bold :
reguler,
          color: currentIndex == 2 ? kPrimaryColor :
kGrayColor,
        ),
      ),
    ],
  ),
),
```

```

        ),
    );
},
);
}

Future<void> _pullToRefresh() async {
    context.read<HealthBloc>().add(HealthGetLandingPage());
}
}

```

19. sign_in_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_button.dart';
import 'package:healio/widget/custom_text_form.dart';

import '../blocs/auth/auth_bloc.dart';
import '../shared/shared_methods.dart';

class SignInPage extends StatefulWidget {
    const SignInPage({Key? key}) : super(key: key);

    @override
    State<SignInPage> createState() => _SignInPageState();
}

final emailController = TextEditingController(text: 'julianreza@gmail.com');
final passwordController = TextEditingController(text: 'julianreza');

class _SignInPageState extends State<SignInPage> {
    @override
    void initState() {
        super.initState();
    }

    @override
    Widget build(BuildContext context) {

```

```
bool validate() {
    if      (emailController.text.isEmpty) || passwordController.text.isEmpty) {
        return false;
    }
    return true;
}

Widget header() {
    return Container(
        margin: EdgeInsets.all(
            defaultMargin,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Text(
                    'Masuk',
                    style: blackTextStyle.copyWith(
                        fontSize: 24,
                        fontWeight: bold,
                    ),
                ),
                Text(
                    'Masuk untuk memantau kondisi kesehatanmu.',
                    style: grayTextStyle,
                ),
            ],
        ),
    );
}

Widget form() {
    return Container(
        margin: EdgeInsets.symmetric(
            horizontal: defaultMargin,
        ),
        child: Column(
            children: [
                CustomTextFormField(
                    label: 'Email',
                    icon: 'assets/fi_mail.png',
                    controller: emailController,
                ),
                const SizedBox(height: 10),
            ],
        ),
    );
}
```

```

CustomTextFormField(
    label: 'Password',
    icon: 'assets/fi_lock.png',
    isPassword: true,
    obscureText: true,
    controller: passwordController,
),
const SizedBox(height: 24),
BlocConsumer<AuthBloc, AuthState>(
    listener: (context, state) {
        if (state is AuthSuccess) {
            Navigator.pushNamedAndRemoveUntil(
                context, '/main', (route) => false);
        } else if (state is AuthFailed) {
            customShowSnackBar(context, state.e);
        }
    },
    builder: (context, state) {
        if (state is AuthLoading) {
            return CustomButton(
                onPressed: () {},
                child: SizedBox(
                    height: 21,
                    width: 21,
                    child: CircularProgressIndicator(
                        color: kWhiteColor,
                    ),
                ),
            );
        }
        return CustomButton(
            onPressed: () {
                if (validate()) {
                    context.read<AuthBloc>().add(
                        AuthSignIn(
                            emailController.text,
                            passwordController.text,
                        ),
                    );
                } else {
                    customShowSnackBar(context, 'Semua field harus
diisi');
                }
            },
            child: Text(

```

```

        'Masuk',
        style: whiteTextStyle.copyWith(
            fontWeight: bold,
        ),
    ),
),
);
},
),
const SizedBox(height: 24),
Wrap(
    mainAxisAlignment: WrapCrossAlignment.center,
    children: [
        Text(
            'Belum punya akun? ',
            style: grayTextStyle,
        ),
        GestureDetector(
            onTap: () {
                Navigator.pushNamed(context, '/sign-up');
            },
            child: Text(
                'Daftar',
                style: primaryTextStyle,
            ),
        ),
    ],
),
],
),
),
);
}
}

return Scaffold(
    body: ListView(
        children: [
            header(),
            form(),
        ],
),
);
}
}

```

20. sign_up_page.dart

```
import 'dart:developer';

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/shared/theme.dart';
import 'package:healio/widget/custom_button.dart';
import 'package:healio/widget/custom_text_form_field.dart';

import '../blocs/auth/auth_bloc.dart';
import '../shared/shared_methods.dart';

class SignUpPage extends StatefulWidget {
    const SignUpPage({Key? key}) : super(key: key);

    @override
    State<SignUpPage> createState() => _SignUpPageState();
}

String gender = 'male';

final nameController = TextEditingController(text: '');
final emailController = TextEditingController(text: '');
final passwordController = TextEditingController(text: '');
String selectedDate = '';

class _SignUpPageState extends State<SignUpPage> {
    DateTime initialDate = DateTime.now();
    @override
    Widget build(BuildContext context) {
        bool validate() {
            if (emailController.text.isEmpty ||
                passwordController.text.isEmpty ||
                nameController.text.isEmpty ||
                selectedDate == '' ||
                gender == '') {
                return false;
            }
            return true;
        }

        Widget header() {
            return Container(

```

```
margin: EdgeInsets.all(
    defaultMargin,
),
child: Column(
    mainAxisAlignment: MainAxisAlignment.start,
    children: [
        Text(
            'Daftar',
            style: blackTextStyle.copyWith(
                fontSize: 24,
                fontWeight: bold,
            ),
        ),
        Text(
            'Daftar untuk memantau kondisi kesehatanmu.',
            style: grayTextStyle,
        ),
    ],
),
);
}

Widget form() {
    return Container(
        margin: EdgeInsets.symmetric(
            horizontal: defaultMargin,
        ),
        child: Column(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                CustomTextField(
                    label: 'Nama',
                    icon: 'assets/fi_user.png',
                    controller: nameController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Email',
                    icon: 'assets/fi_mail.png',
                    controller: emailController,
                ),
                const SizedBox(height: 16),
                CustomTextField(
                    label: 'Password',
                    icon: 'assets/fi_lock.png',
                ),
            ],
        ),
    );
}
```

```

        isPassword: true,
        obscureText: true,
        controller: passwordController,
    ),
    const SizedBox(height: 16),
    Text(
        'Jenis Kelamin',
        style: blackTextStyle.copyWith(
            fontSize: 16,
            fontWeight: bold,
        ),
    ),
    const SizedBox(height: 10),
    Row(
        children: [
            Expanded(
                child: SizedBox(
                    height: 46,
                    child: ElevatedButton(
                        onPressed: () {
                            setState(() {
                                gender = 'male';
                            });
                        },
                        style: ElevatedButton.styleFrom(
                            elevation: 0,
                            primary:
                                gender == 'male' ? kPrimaryColor :
kLightGrayColor,
                            shape: RoundedRectangleBorder(
                                borderRadius: BorderRadius.circular(
                                    defaultRadius,
                                ),
                            ),
                        ),
                    ),
                ),
            ),
            child: Wrap(
                crossAxisAlignment:
WrapCrossAlignment.center,
                children: [
                    Icon(
                        Icons.male,
                        size: 20,
                        color: gender == 'male' ? kWhiteColor :
kGrayColor,
                ),
            ],
        ),
    ),

```

```
        const SizedBox(
            width: 10,
        ),
        Text(
            'Laki-laki',
            style: whiteTextStyle.copyWith(
                color:
                    gender == 'male' ? kWhiteColor :
kGrayColor,
            ),
            ),
            ],
            ),
        ),
        ),
    ),
),
const SizedBox(width: 16),
Expanded(
    child: SizedBox(
        height: 46,
        child: ElevatedButton(
            onPressed: () {
                setState(() {
                    gender = 'female';
                });
            },
            style: ElevatedButton.styleFrom(
                elevation: 0,
                primary:
                    gender == 'female' ? kRedColor :
kLightGrayColor,
                shape: RoundedRectangleBorder(
                    borderRadius: BorderRadius.circular(
                        defaultRadius,
                    ),
                    ),
                ),
                child: Wrap(
                    crossAxisAlignment:
WrapCrossAlignment.center,
                    children: [
                    Icon(
                        Icons.female,
                        size: 20,
                        color:
```

```
        gender == 'female' ? kWhiteColor :  
kGrayColor,  
        ),  
        const SizedBox(  
            width: 10,  
        ),  
        Text(  
            'Perempuan',  
            style: whiteTextStyle.copyWith(  
                color:  
                    gender == 'female' ? kWhiteColor :  
kGrayColor,  
                ),  
                ),  
                ],  
                ),  
                ),  
                ),  
                ),  
                ],  
            ),  
            const SizedBox(height: 16),  
            Text(  
                'Tanggal Lahir',  
                style: blackTextStyle.copyWith(  
                    fontSize: 16,  
                    fontWeight: bold,  
                ),  
            ),  
            const SizedBox(height: 10),  
            GestureDetector(  
                onTap: () {  
                    showDatePicker(  
                        context: context,  
                        initialDate: initialDate,  
                        firstDate: DateTime(1950),  
                        lastDate: DateTime(2050),  
                    ).then((value) {  
                        if (value != null) {  
                            setState(() {  
                                selectedDate = value.toString();  
                            });  
                        }  
                    });  
                },  
            ),  
        ),  
    ),  
},
```

```
        child: Container(
            height: 57,
            decoration: BoxDecoration(
                color: kLightGrayColor,
                borderRadius: BorderRadius.circular(
                    defaultRadius,
                ),
            ),
            padding: const EdgeInsets.symmetric(
                horizontal: 16,
            ),
            child: Row(
                children: [
                    Image.asset(
                        'assets/fi_trello.png',
                        height: 16,
                        color: kGrayColor,
                    ),
                    const SizedBox(width: 10),
                    Text(
                        selectedDate,
                        style: grayTextStyle,
                    ),
                ],
            ),
        ),
        const SizedBox(height: 16),
        BlocConsumer<AuthBloc, AuthState>(
            listener: (context, state) {
                if (state is AuthSuccess) {
                    Navigator.pushNamedAndRemoveUntil(
                        context, '/main', (route) => false);
                } else if (state is AuthFailed) {
                    customShowSnackBar(context, state.e);
                }
            },
            builder: (context, state) {
                if (state is AuthLoading) {
                    return CustomButton(
                        onPressed: () {},
                        child: SizedBox(
                            height: 21,
                            width: 21,
                            child: CircularProgressIndicator(

```

```
        color: kWhiteColor,
    ),
),
);
}
return CustomButton(
 onPressed: () {
if (validate()) {
context.read<AuthBloc>().add(
AuthSignUp(
emailController.text,
passwordController.text,
nameController.text,
gender,
selectedDate,
),
);
} else {
customShowSnackBar(context, 'Semua field harus
diisi');
}
},
child: Text(
'Daftar',
style: whiteTextStyle.copyWith(
fontWeight: bold,
),
),
),
),
),
),
),
const SizedBox(height: 24),
Center(
child: Wrap(
crossAxisAlignment: WrapCrossAlignment.center,
children: [
Text(
'Sudah punya akun? ',
style: grayTextStyle,
),
GestureDetector(
onTap: () {
Navigator.pushNamed(context, '/sign-in');
},
child: Text(
```

```

        'Masuk',
        style: primaryTextStyle,
    ),
),
],
),
),
],
),
);
}

return Scaffold(
body: ListView(
children: [
header(),
form(),
],
),
);
}
}

```

21. splash_page.dart

```

import 'dart:async';

import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:heilio/shared/theme.dart';

import '../blocs/auth/auth_bloc.dart';

class SplashPage extends StatefulWidget {
const SplashPage({Key? key}) : super(key: key);

@Override
State<SplashPage> createState() => _SplashPageState();
}

class _SplashPageState extends State<SplashPage> {

```

```

@Override
void initState() {
    Timer(const Duration(seconds: 2), () {
        User? user = FirebaseAuth.instance.currentUser;

        if (user == null) {
            Navigator.pushNamedAndRemoveUntil(
                context, '/sign-in', (route) => false);
        } else {
            context.read<AuthBloc>().add(AuthGetCurrentUser(user.uid));
            Navigator.pushNamedAndRemoveUntil(context, '/main', (route)
=> false);
        }
    });
    super.initState();
}

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: Center(
            child: Text(
                'healio',
                style: primaryTextStyle.copyWith(
                    fontSize: 56,
                    fontWeight: bold,
                ),
            ),
        ),
    );
}

```

22. user_page.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/blocs/page/page_cubit.dart';
import 'package:healio/pages/edit_date_of_birth_page.dart';
import 'package:healio/pages/edit_gender_page.dart';
import 'package:healio/pages/edit_name_page.dart';
import 'package:healio/shared/shared_methods.dart';

```

```
import 'package:heilio/shared/theme.dart';
import 'package:heilio/widget/custom_button.dart';
import 'package:heilio/widget/user_about_item.dart';
import 'package:health/health.dart';
import 'package:intl/intl.dart';

import '../blocs/auth/auth_bloc.dart';

class UserPage extends StatelessWidget {
    const UserPage({Key? key}) : super(key: key);

    @override
    Widget build(BuildContext context) {
        Widget header() {
            return Container(
                margin: EdgeInsets.all(
                    defaultMargin,
                ),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.start,
                    children: [
                        Text(
                            'Akun',
                            style: blackTextStyle.copyWith(
                                fontSize: 26,
                                fontWeight: bold,
                            ),
                        ),
                        const SizedBox(height: 5),
                        Text(
                            'Informasi terkait akun kamu',
                            style: grayTextStyle,
                        ),
                    ],
                ),
            );
        }

        Widget body() {
            return BlocBuilder<AuthBloc, AuthState>(
                builder: (context, state) {
                    if (state is AuthSuccess) {
                        final date = DateTime.parse(state.user.dateOfBirth);
                        String formattedDate = DateFormat('y-MM-d').format(date);
                    }
                },
            );
        }
    }
}
```

```
        return Container(
            margin: EdgeInsets.symmetric(
                horizontal: defaultMargin,
            ),
            child: Column(
                mainAxisAlignment: CrossAxisAlignment.start,
                children: [
                    Center(
                        child: Column(
                            children: [
                                Container(
                                    height: 84,
                                    width: 84,
                                    margin: const EdgeInsets.only(bottom: 10),
                                    decoration: const BoxDecoration(
                                        image: DecorationImage(
                                            image: AssetImage(
                                                'assets/avatar.png',
                                            ),
                                            fit: BoxFit.cover,
                                        ),
                                        shape: BoxShape.circle,
                                    ),
                                ),
                                Column(
                                    crossAxisAlignment:
CrossAxisAlignment.center,
                                    children: [
                                        Text(
                                            state.user.name,
                                            style: blackTextStyle.copyWith(
                                                fontSize: 24,
                                                fontWeight: bold,
                                            ),
                                        ),
                                        Text(
                                            state.user.email,
                                            style: grayTextStyle,
                                        ),
                                    ],
                                ),
                            ],
                        ),
                    ),
                    SizedBox(
```

```
        height: defaultMargin,
    ),
    Text(
        'Tentang',
        style: blackTextStyle.copyWith(
            fontSize: 16,
            fontWeight: bold,
        ),
    ),
    const SizedBox(height: 10),
    Container(
        padding: const EdgeInsets.symmetric(vertical:
10),
        decoration: BoxDecoration(
            color: kLightGrayColor,
            borderRadius: BorderRadius.circular(
                defaultRadius,
            ),
        ),
        margin: EdgeInsets.only(
            bottom: defaultMargin,
        ),
        child: Column(
            children: [
                UserAboutItem(
                    label: 'Nama',
                    value: state.user.name,
                    page: EditNamePage(
                        name: state.user.name,
                    ),
                ),
                UserAboutItem(
                    label: 'Jenis Kelamin',
                    value: state.user.gender == 'male'
                        ? 'Laki-laki'
                        : 'Perempuan',
                    page: const EditGenderPage(),
                ),
                UserAboutItem(
                    label: 'Tanggal Lahir',
                    value: formattedDate,
                    page: EditDateOfBirthPage(
                        date: state.user.dateOfBirth,
                    ),
                ),
            ],
        ),
    ),

```

```

        ],
    ),
),
),
const SizedBox(
    height: 10,
),
BlocConsumer<AuthBloc, AuthState>(
    listener: (context, state) {
        if (state is AuthFailed) {
            customShowSnackBar(context, state.e);
        } else if (state is AuthInitial) {
            Navigator.pushNamedAndRemoveUntil(
                context, '/sign-in', (route) => false);
        }
    },
    builder: (context, state) {
        if (state is AuthLoading) {
            return CustomButton(
                onPressed: () {},
                child: SizedBox(
                    height: 21,
                    width: 21,
                    child: CircularProgressIndicator(
                        color: kWhiteColor,
                    ),
                ),
            );
        }
        return CustomButton(
            onPressed: () {

context.read<AuthBloc>().add(AuthSignOut());
            context.read<PageCubit>().setPage(0);
            Navigator.pushNamedAndRemoveUntil(
                context, '/sign-in', (route) =>
false);
        },
        child: Wrap(
            crossAxisAlignment:
WrapCrossAlignment.center,
            children: [
                Image.asset(
                    'assets/fi_log-out.png',
                    height: 16,
                    color: kWhiteColor,

```

```

        ),
        const SizedBox(width: 10),
        Text(
            'Logout',
            style: whiteTextStyle.copyWith(
                fontWeight: bold,
            ),
            ),
            ],
            ),
            ),
            );
        },
        ],
        ),
        );
    }
    return Container();
},
);
}
}

return Scaffold(
body: ListView(
children: [
header(),
body(),
],
),
);
}
}

```

23. webview_page.dart

```

import 'dart:io';

import 'package:flutter/material.dart';
import 'package:webview_flutter/webview_flutter.dart';

import '../shared/theme.dart';

```

```
class WebviewPage extends StatefulWidget {
  const WebviewPage({required this.url, Key? key}) : super(key: key);

  final String url;

  @override
  State<WebviewPage> createState() => _WebviewPageState();
}

class _WebviewPageState extends State<WebviewPage> {
  @override
  void initState() {
    super.initState();
    if (Platform.isAndroid) WebView.platform = AndroidWebView();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Column(
          children: [
            GestureDetector(
              onTap: () {
                Navigator.pop(context);
              },
              child: Container(
                width: double.infinity,
                padding: const EdgeInsets.all(16),
                color: kPrimaryColor,
                child: Row(
                  children: [
                    Icon(
                      Icons.chevron_left,
                      size: 24,
                      color: kWhiteColor,
                    ),
                    const SizedBox(
                      width: 10,
                    ),
                    Text(
                      'Kembali ke aplikasi',
                      style: whiteTextStyle.copyWith(
                        fontSize: 14,

```

```

        fontWeight: bold,
    ),
),
],
),
),
),
),
),
),
),
),
),
),
),
),
),
),
),
),
),
);
}
}

```

24. article_card.dart

```

import 'package:flutter/material.dart';
import 'package:healio/models/article_model.dart';
import '../pages/webview_page.dart';
import '../shared/theme.dart';

class ArticleCard extends StatelessWidget {
    const ArticleCard(
        this.article, {
        Key? key,
    }) : super(key: key);

    final ArticleModel article;

    @override
    Widget build(BuildContext context) {
        return Container(
            margin: const EdgeInsets.only(top: 10),
            decoration: BoxDecoration(
                boxShadow: [
                    BoxShadow(
                        color: kGrayColor.withOpacity(0.1),

```

```
        offset: const Offset(3, 3),
        blurRadius: 10,
    ),
],
),
width: double.infinity,
child: ClipRRect(
    borderRadius: BorderRadius.circular(12),
    child: GestureDetector(
        onTap: () {
            Navigator.push(
                context,
                MaterialPageRoute(
                    builder: (context) => WebviewPage(
                        url: article.link.toString(),
                    ),
                ),
            );
        },
        child: Container(
            decoration: BoxDecoration(
                color: kwhiteColor,
                border: Border.all(
                    color: kLightGrayColor,
                ),
            ),
        ),
        child: Row(
            mainAxisAlignment: MainAxisAlignment.start,
            children: [
                Container(
                    height: 96,
                    width: 144,
                    decoration: BoxDecoration(
                        image: DecorationImage(
                            image: NetworkImage(
                                article.thumbnail.toString(),
                            ),
                            fit: BoxFit.cover,
                        ),
                ),
            ],
        ),
        Expanded(
            child: Container(
                padding: const EdgeInsets.all(10),
                child: Column(

```

```
        crossAxisAlignment: CrossAxisAlignment.start,
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Text(
            article.title.toString(),
            style: blackTextStyle.copyWith(
              fontWeight: bold,
            ),
          ),
          const SizedBox(
            height: 5,
          ),
          Text(
            article.description!,
            style: grayTextStyle.copyWith(
              fontSize: 14,
            ),
            maxLines: 1,
            overflow: TextOverflow.ellipsis,
          ),
        ],
      ),
    ),
  ],
),
),
],
),
),
),
),
),
),
),
),
),
),
),
),
),
),
);
}
}
```

25. custom_app_bar.dart

```
import 'package:flutter/material.dart';

import '../shared/theme.dart';

class CustomAppBar extends StatelessWidget {
  const CustomAppBar({
    required this.title,
```

```
this.action = const SizedBox(),
Key? key,
}) : super(key: key);

final String title;
final Widget? action;

@Override
Widget build(BuildContext context) {
    return Container(
        margin: EdgeInsets.all(
            defaultMargin,
        ),
        child: Row(
            children: [
                Row(
                    children: [
                        GestureDetector(
                            onTap: () {
                                Navigator.pop(context);
                            },
                            child: Container(
                                padding: const EdgeInsets.all(8),
                                decoration: BoxDecoration(
                                    color: kLightGrayColor,
                                    borderRadius: BorderRadius.circular(
                                        defaultRadius,
                                    ),
                                ),
                            ),
                        ),
                        child: const Icon(
                            Icons.chevron_left,
                            size: 24,
                        ),
                    ],
                ),
                const SizedBox(width: 10),
                Text(
                    title,
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                        fontWeight: bold,
                    ),
                )
            ],
        ),
    ),
}
```

```
        const Spacer(),
        SizedBox(
            child: action,
        ),
    ],
),
);
}
}
```

26. custom_button.dart

```
import 'package:flutter/material.dart';

import '../shared/theme.dart';

class CustomButton extends StatelessWidget {
    const CustomButton({
        required this.onPressed,
        required this.child,
        Key? key,
    }) : super(key: key);

    final Widget child;
    final Function() onPressed;

    @override
    Widget build(BuildContext context) {
        return SizedBox(
            height: 57,
            width: double.infinity,
            child: ElevatedButton(
                onPressed: () {
                    onPressed();
                },
                style: ElevatedButton.styleFrom(
                    primary: kPrimaryColor,
                    shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(
                            defaultRadius,
                        ),
                    ),
                ),
            ),
        );
    }
}
```

```
        ),
        child: child,
    ),
);
}
}
```

27. custom_text_form_field.dart

```
import 'package:flutter/material.dart';

import '../shared/theme.dart';

class CustomTextField extends StatefulWidget {
    const CustomTextField({
        required this.label,
        this.icon,
        this.isPassword = false,
        this.obscureText = false,
        required this.controller,
        this.hint,
        Key? key,
    }) : super(key: key);

    final String label;
    final bool isPassword;
    final bool obscureText;
    final String? icon;
    final TextEditingController controller;
    final String? hint;

    @override
        State<CustomTextField>          createState()      =>
    _CustomTextFieldState();
}

bool isShow = false;

class _CustomTextFieldState extends State<CustomTextField> {
    @override
    void initState() {
        super.initState();
```

```
        isShow = widget.obscureText;
    }

    @override
    Widget build(BuildContext context) {
        return Column(
            mainAxisAlignment: CrossAxisAlignment.start,
            children: [
                Text(
                    widget.label,
                    style: blackTextStyle.copyWith(
                        fontSize: 16,
                        fontWeight: bold,
                    ),
                ),
                const SizedBox(
                    height: 10,
                ),
                Container(
                    height: 57,
                    decoration: BoxDecoration(
                        color: kLightGrayColor,
                        borderRadius: BorderRadius.circular(
                            defaultRadius,
                        ),
                    ),
                ),
                padding: const EdgeInsets.symmetric(
                    horizontal: 16,
                    vertical: 10,
                ),
                child: Row(
                    children: [
                        widget.icon != null
                            ? Row(
                                children: [
                                    Image.asset(
                                        widget.icon!,
                                        height: 16,
                                        color: kGrayColor,
                                    ),
                                    const SizedBox(width: 10),
                                ],
                            )
                            : const SizedBox(),
                ],
            ),
        );
    }
}
```

```

        child: TextFormField(
            controller: widget.controller,
            decoration: InputDecoration.collapsed(
                hintText: widget.hint ?? 'Masukan
${widget.label}',
                hintStyle: grayTextStyle,
            ),
            obscureText: isShow ? true : false,
        ),
    ),
    widget.isPassword
        ? GestureDetector(
            onTap: () {
                setState(() {
                    isShow = !isShow;
                });
            },
            child: Container(
                margin: const EdgeInsets.only(left: 10),
                child: Image.asset(
                    isShow
                        ? 'assets/fi_eye.png'
                        : 'assets/fi_eye-off.png',
                    height: 16,
                    color: kGrayColor,
                ),
            ),
        )
        :
        const SizedBox(),
    ],
),
),
],
),
);
}
}

```

28. health_category_tile.dart

```

import 'package:flutter/material.dart';
import 'package:healio/shared/theme.dart';

```

```
enum HealthCategory {
    heartrate,
    bloodoxygen,
}

class HealthCategoryTile extends StatelessWidget {
    const HealthCategoryTile({
        required this.healthCategory,
        Key? key,
    }) : super(key: key);

    final HealthCategory healthCategory;

    @override
    Widget build(BuildContext context) {
        return GestureDetector(
            onTap: () {
                Navigator.pushNamed(
                    context,
                    healthCategory == HealthCategory.heartrate
                        ? '/heart-rate'
                        : '/blood-oxygen',
                );
            },
            child: Container(
                padding: const EdgeInsets.symmetric(
                    horizontal: 16,
                    vertical: 16,
                ),
                decoration: BoxDecoration(
                    color: kLightGrayColor,
                    borderRadius: BorderRadius.circular(
                        12,
                    ),
                ),
                child: Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                        Text(
                            healthCategory == HealthCategory.heartrate
                                ? 'Denyut Jantung'
                                : 'Saturasi Oksigen',
                            style: blackTextStyle,
                        ),
                        const Spacer(),
                    ],
                ),
            ),
        );
    }
}
```

```

        Container(
            height: 50,
            width: 50,
            decoration: BoxDecoration(
                color: kWhiteColor,
                shape: BoxShape.circle,
            ),
            padding: const EdgeInsets.all(13),
            child: Image.asset(
                healthCategory == HealthCategory.heartrate
                    ? 'assets/fi_heart.png'
                    : 'assets/fi_droplet.png',
                color: healthCategory == HealthCategory.heartrate
                    ? kRedColor
                    : kPrimaryColor,
            ),
        ),
    ],
),
),
);
),
);
);
);
}
}

```

29. health_data_item.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:healio/pages/blood_oxygen_page.dart';
import 'package:health/health.dart';
import 'package:intl/intl.dart';
import 'package:healio/shared/shared_methods.dart';

import '../blocs/auth/auth_bloc.dart';
import '../shared/theme.dart';

int age = 0;

class HealthDataItem extends StatefulWidget {
    const HealthDataItem(
        this.healthDataPoint, {
        this.isAll = false,
    );
}

```

```

        this.category = 'heart_rate',
        Key? key,
    }) : super(key: key);

    final HealthDataPoint healthDataPoint;
    final bool isAll;
    final String category;

    @override
    State<HealthDataItem> createState() => _HealthDataItemState();
}

class _HealthDataItemState extends State<HealthDataItem> {
    @override
    void initState() {
        super.initState();
        getAge();
    }

    void getAge() {
        final authState = context.read<AuthBloc>().state;
        if (authState is AuthSuccess) {
            calculateAge(DateTime.parse(authState.user.dateOfBirth));
        }
    }

    @override
    Widget build(BuildContext context) {
        String getTodayDate =
            DateFormat("Hm").format(widget.healthDataPoint.dateFrom);
        String getAllDate =
            DateFormat("d MMMM y h:m
aaa").format(widget.healthDataPoint.dateFrom);

        String labelCategory = setCategory(
            getValueFromHealthValue(widget.healthDataPoint.value),
            widget.category,
            age,
        );
    }

    return Container(
        padding: const EdgeInsets.symmetric(
            horizontal: 16,
            vertical: 16,
    );
}

```

```
        ),
        decoration: BoxDecoration(
            color: kLightGrayColor,
            borderRadius: BorderRadius.circular(
                defaultRadius,
            ),
            boxShadow: [
                BoxShadow(
                    color: kLightGrayColor,
                    offset: const Offset(3, 3),
                    blurRadius: 10,
                ),
            ],
        ),
        margin: const EdgeInsets.only(
            top: 10,
        ),
        child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
                Column(
                    mainAxisAlignment: MainAxisAlignment.start,
                    children: [
                        Wrap(
                            mainAxisAlignment: WrapCrossAlignment.end,
                            children: [
                                Text(
                                    '${getValueFromHealthValue(widget.healthDataPoint.value)}',
                                    style: blackTextStyle.copyWith(
                                        fontSize: 20,
                                        fontWeight: bold,
                                    ),
                                ),
                                Text(
                                    widget.category == 'heart_rate' ? 'BPM' : '%',
                                    style: grayTextStyle.copyWith(
                                        fontSize: widget.category == 'heart_rate' ? 14
                                         : 20,
                                    ),
                                ),
                            ],
                ),
                const SizedBox(height: 5),
            ],
        ),
    
```

```

        padding: const EdgeInsets.symmetric(
            horizontal: 8,
            vertical: 3,
        ),
        decoration: BoxDecoration(
            color: (labelCategory == 'normal' ? kGreenColor :
kRedColor)
                .withOpacity(0.1),
            borderRadius: BorderRadius.circular(5),
        ),
        child: Text(
            labelCategory,
            style: greenTextStyle.copyWith(
                fontSize: 12,
                color: labelCategory == 'normal' ? kGreenColor :
kRedColor,
            ),
        ),
    ),
),
],
),
const Spacer(),
Text(
    widget.isAll ? getAllDate : getTodayDate,
    style: grayTextStyle,
),
],
),
);
}
}

```

30. more_button.dart

```

import 'package:flutter/material.dart';

import '../shared/theme.dart';

class MoreButton extends StatelessWidget {
    const MoreButton({
        required this.text,
        required this.route,
    })
}

```

```
Key? key,
}) : super(key: key);

final String text;
final String route;

@Override
Widget build(BuildContext context) {
    return GestureDetector(
        onTap: () {
            Navigator.pushNamed(context, route);
        },
        child: Container(
            margin: EdgeInsets.only(
                left: defaultMargin,
                right: defaultMargin,
                top: 10,
            ),
            padding: const EdgeInsets.all(16),
            decoration: BoxDecoration(
                color: kLightGrayColor,
                borderRadius: BorderRadius.circular(
                    defaultRadius,
                ),
            ),
            child: Row(
                children: [
                    Text(
                        text,
                        style: blackTextStyle.copyWith(
                            fontSize: 14,
                        ),
                    ),
                    const Spacer(),
                    const Icon(
                        Icons.chevron_right,
                    ),
                ],
            ),
        );
    }
}
```

31. user_about_item.dart

```
import 'package:flutter/material.dart';

import '../shared/theme.dart';

class UserAboutItem extends StatelessWidget {
  const UserAboutItem({
    required this.label,
    required this.value,
    required this.page,
    Key? key,
  }) : super(key: key);

  final String label;
  final String value;
  final Widget page;
  @override
  Widget build(BuildContext context) {
    return Container(
      padding: const EdgeInsets.symmetric(
        vertical: 10,
        horizontal: 16,
      ),
      child: Row(
        children: [
          Text(
            label,
            style: blackTextStyle,
          ),
          const Spacer(),
          GestureDetector(
            onTap: () {
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => page,
                ),
              );
            },
            child: Wrap(
              crossAxisAlignment: WrapCrossAlignment.center,
```

```

        children: [
          Text(
            value,
            style: grayTextStyle,
          ),
          const SizedBox(width: 5),
          Icon(
            Icons chevron_right,
            color: kGrayColor,
          ),
        ],
      ),
    ],
  );
}
}

```

32. shared_methods.dart

```

import 'package:another_flushbar/flushbar.dart';
import 'package:flutter/cupertino.dart';
import 'package:healio/shared/theme.dart';
import 'package:health/health.dart';

int getValueFromHealthValue(HealthValue healthValue) {
  List<String> value = healthValue.toString().split('.');
  return int.parse(value[0]);
}

void customShowSnackBar(BuildContext context, String message) {
  Flushbar(
    flushbarPosition: FlushbarPosition.TOP,
    message: message,
    backgroundColor: kRedColor,
    duration: const Duration(seconds: 2),
  ).show(context);
}

String setCategory(
  int value,

```

```

    String category,
    int age,
) {
    String label = "";

    if (category == 'heart_rate') {
        if (age <= 1) {
            if (value < 70) {
                label = 'rendah';
            } else if (value >= 70 && value < 190) {
                label = 'normal';
            } else {
                label = 'tinggi';
            }
        } else if (age > 1 && age <= 10) {
            if (value < 80) {
                label = 'rendah';
            } else if (value >= 80 && value < 110) {
                label = 'normal';
            } else {
                label = 'tinggi';
            }
        } else if (age > 10) {
            if (value < 60) {
                label = 'rendah';
            } else if (value >= 60 && value < 100) {
                label = 'normal';
            } else {
                label = 'tinggi';
            }
        }
    } else if (category == 'blood_oxygen') {
        if (value < 95) {
            label = 'rendah';
        } else {
            label = 'normal';
        }
    }

    return label;
}

calculateAge(DateTime dateOfBirth) {
    DateTime currentDate = DateTime.now();
    int age = currentDate.year - dateOfBirth.year;
}

```

```

int month1 = currentDate.month;
int month2 = dateOfBirth.month;
if (month2 > month1) {
    age--;
} else if (month1 == month2) {
    int day1 = currentDate.day;
    int day2 = dateOfBirth.day;
    if (day2 > day1) {
        age--;
    }
}
return age;
}

```

33. theme.dart

```

import 'package:flutter/material.dart';
import 'package:google_fonts/google_fonts.dart';

double defaultMargin = 24.0;
double defaultRadius = 12;

Color kGrayColor = const Color(0xff9B9B9B);
Color kBlackColor = const Color(0xff252525);
Color kPrimaryColor = const Color(0xff0572f1);
Color kLightGrayColor = const Color(0xfff3f4f8);
Color kSecondaryColor = const Color(0xffB6C1EA);
Color kWhiteColor = const Color(0xfffffdfd);
Color kTransparentColor = Colors.transparent;
Color kRedColor = const Color(0xFFFF2566);
Color kGreenColor = const Color(0xff2ecc71);

TextStyle blackTextStyle = GoogleFonts.nunito(
    color: kBlackColor,
);
TextStyle primaryTextStyle = GoogleFonts.nunito(
    color: kPrimaryColor,
);
TextStyle grayTextStyle = GoogleFonts.nunito(
    color: kGrayColor,
);
TextStyle whiteTextStyle = GoogleFonts.nunito(

```

```

    color: kWhiteColor,
);
TextStyle lightGrayTextStyle = GoogleFonts.nunito(
    color: kLightGrayColor,
);
TextStyle redTextStyle = GoogleFonts.nunito(
    color: kRedColor,
);
TextStyle greenTextStyle = GoogleFonts.nunito(
    color: kGreenColor,
);

FontWeight light = FontWeight.w300;
FontWeight reguler = FontWeight.w400;
FontWeight medium = FontWeight.w500;
FontWeight semiBold = FontWeight.w600;
FontWeight bold = FontWeight.w700;
FontWeight extraBold = FontWeight.w800;
FontWeight black = FontWeight.w900;

```

34. article_model.dart

```

class ArticleModel {
    String? title;
    String? description;
    String? link;
    String? thumbnail;

    ArticleModel({
        this.title = '',
        this.description = '',
        this.link = '',
        this.thumbnail = '',
    });

    factory ArticleModel.fromJson(Map<String, dynamic> json) =>
    ArticleModel(
        title: json['title'],
        description: json['description'],
        link: json['link'],
        thumbnail: json['thumbnail'],
    );
}

```

```
}
```

35. health_model.dart

```
class HealthModel {  
    final String type;  
    final int value;  
    final String unit;  
    final String dateFrom;  
  
    HealthModel({  
        required this.type,  
        required this.value,  
        required this.unit,  
        required this.dateFrom,  
    });  
  
    factory HealthModel.fromJson(Map<String, dynamic> json) =>  
    HealthModel(  
        type: json['type'],  
        value: json['value'],  
        unit: json['unit'],  
        dateFrom: json['dateFrom'],  
    );  
  
    Map<String, dynamic> toJson() => {  
        'type': type,  
        'value': value,  
        'unit': unit,  
        'dateFrom': dateFrom,  
    };  
}
```

36. heart_disease_form_model.dart

```
class HeartDiseaseFormModel {  
    final int? age;  
    final int? sex;  
    final int? cigs;  
    final int? chol;  
    final int? sBP;
```

```

final int? dia;
final int? dBp;
final int? gluc;
final int? hRate;

HeartDiseaseFormModel({
  this.age,
  this.sex,
  this.cigs,
  this.chol,
  this.sBP,
  this.dia,
  this.dBp,
  this.gluc,
  this.hRate,
});

Map<String, dynamic> toJson() => {
  'age': age,
  'sex': sex,
  'cigs': cigs,
  'chol': chol,
  'sBP': sBP,
  'dia': dia,
  'dBp': dBp,
  'gluc': gluc,
  'hRate': hRate,
};

}

```

37. user_model.dart

```

class UserModel {
  final String id;
  final String email;
  final String name;
  final String gender;
  final String dateOfBirth;

  UserModel(
    {required this.id,
    required this.email,
    
```

```

        required this.name,
        required this.gender,
        required this.dateOfBirth));

factory UserModel.fromJson(String id, Map<String, dynamic> json)
=> UserModel(
    id: id,
    email: json['email'],
    name: json['name'],
    gender: json['gender'],
    dateOfBirth: json['date_of_birth'],
);

Map<String, dynamic> toJson() => {
    'id': id,
    'email': email,
    'name': name,
    'gender': gender,
    'date_of_birth': dateOfBirth,
};

}

```

38. auth_service.dart

```

import 'package:firebase_auth/firebase_auth.dart';
import 'package:healio/services/user_service.dart';

import '../models/user_model.dart';

class AuthService {
    final FirebaseAuth _auth = FirebaseAuth.instance;

    Future<UserModel> signIn({
        required String email,
        required String password,
    }) async {
        try {
            UserCredential userCredential = await
            _auth.signInWithEmailAndPassword(
                email: email,
                password: password,
            );
        }
    }
}

```

```

UserModel user =
    await UserService().getUserById(userCredential.user!.uid);
return user;
} catch (e) {
    rethrow;
}
}

Future<UserModel> signUp({
    required String email,
    required String password,
    required String name,
    required String gender,
    required String dateOfBirth,
}) async {
    try {
        UserCredential userCredential = await _auth
            .createUserWithEmailAndPassword(email: email, password:
password);
        UserModel user = UserModel(
            id: userCredential.user!.uid,
            email: email,
            name: name,
            gender: gender,
            dateOfBirth: dateOfBirth,
        );
        await UserService().setUser(user);
        return user;
    } catch (e) {
        rethrow;
    }
}

Future<void> signOut() async {
    try {
        await _auth.signOut();
    } catch (e) {
        rethrow;
    }
}
}

```

39. health_service.dart

```
import 'dart:developer';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:health/health.dart';
import 'package:permission_handler/permission_handler.dart';

import '../models/health_model.dart';

class HealthService {
    final CollectionReference _healthReference =
        FirebaseFirestore.instance.collection('health');

    Future<List<HealthDataPoint>> fetchData(
        DateTime now,
        DateTime yesterday,
        List<HealthDataType> types,
    ) async {
        List<HealthDataPoint> healthDataList = [];
        HealthFactory health = HealthFactory();

        final permissions = [
            HealthDataAccess.READ,
        ];

        bool requested =
            await health.requestAuthorization(types, permissions: permissions);

        await Permission.activityRecognition.request();
        await Permission.location.request();

        if (requested) {
            try {
                List<HealthDataPoint> healthData =
                    await health.getHealthDataFromTypes(yesterday, now,
                types);
                healthDataList.addAll((healthData.length < 100)
                    ? healthData
                    : healthData.sublist(0, 100));
            } catch (error) {
        }
    }
}
```

```

        throw Exception("Exception in getHealthDataFromTypes:
$error");
    }

                healthDataList =
HealthFactory.removeDuplicates(healthDataList);

    healthDataList.sort((a, b) {
        var aDate = a.dateFrom;
        var bDate = b.dateFrom;
        return bDate.compareTo(aDate);
    });

    return healthDataList;
} else {
    throw Exception("Authorization not granted");
}
}

Future<HealthModel> getHeartRateFromFirebase() async {
try {
    DocumentSnapshot snapshot =
        await _healthReference.doc('heart_rate').get();

    return HealthModel(
        type: snapshot['type'],
        value: snapshot['value'],
        unit: snapshot['unit'],
        dateFrom: snapshot['dateFrom'],
    );
} catch (e) {
    throw Exception(e);
}
}

Future<void> updateHeartRateFromFirebase(
    int value,
    String dateFrom,
) async {
try {
    _healthReference.doc('heart_rate').update({
        'value': value,
        'dateFrom': dateFrom,
    });
} catch (e) {
}
}

```

```

        throw Exception(e);
    }
}

Future<HealthModel> getBloodOxygenFromFirebase() async {
    try {
        DocumentSnapshot snapshot =
            await _healthReference.doc('blood_oxygen').get();

        return HealthModel(
            type: snapshot['type'],
            value: snapshot['value'],
            unit: snapshot['unit'],
            dateFrom: snapshot['dateFrom'],
        );
    } catch (e) {
        throw Exception(e);
    }
}

Future<void> updateBloodOxygenFromFirebase(
    int value,
    String dateFrom,
) async {
    try {
        _healthReference.doc('blood_oxygen').update({
            'value': value,
            'dateFrom': dateFrom,
        });
    } catch (e) {
        throw Exception(e);
    }
}

```

40. heart_disease_service.dart

```

import 'dart:convert';
import 'dart:developer';

import 'package:healio/models/heart_disease_form_model.dart';
import 'package:http/http.dart' as http;

```

```

class HeartDiseaseService {
    String baseUrl = 'https://heartapi.herokuapp.com';

    Future<Map<String, dynamic>> prediction(HeartDiseaseFormModel
data) async {
        var url = Uri.parse(
            '$baseUrl/predict?age=${data.age}&sex=${data.sex}&cigs=${data.cigs}&
chol=${data.chol}&sBP=${data.sBP}&dia=${data.dia}&dBP=${data.dBP}&gl
uc=${data.gluc}&hRate=${data.hRate}');
        var res = await http.get(url);

        if (res.statusCode == 200) {
            var result = jsonDecode(res.body);

            return result;
        } else {
            throw Exception('Gagal memprediksi');
        }
    }
}

```

41. push_notification_service.dart

```

import 'dart:convert';
import 'package:http/http.dart' as http;

class PushNotificationService {
    String baseUrl = "https://fcm.googleapis.com/fcm/send";

    Future<bool> pushNotification(
        String title,
        String desc,
    ) async {
        var url = Uri.parse(baseUrl);

        var headers = {
            'Content-Type': 'application/json',
            'Authorization':
                'key=AAAAstAiflM:APA91bHcn_64JkornQ-u6KRSZtE-C_n2-

```

```

dHgV7PzqQj7kj83AVAK43AfXGM659FxEjqxH2bQXwE6MbNhzoatFTOAq9kLGjwhMbDtgy5AA2RapIdNhRiMMJHNBr9vTM-6NHSS-c6Vc23x',
};

var body = jsonEncode({
  "to": "/topics/health",
  "collapse_key": "type_a",
  "notification": {
    "title": title,
    "body": desc,
  }
});

var response = await http.post(
  url,
  headers: headers,
  body: body,
);

if (response.statusCode == 200) {
  return true;
} else {
  return false;
}
}
}

```

42. user_service.dart

```

import 'package:cloud_firestore/cloud_firestore.dart';
import '../models/user_model.dart';

class UserService {
  final CollectionReference _userReference =
      FirebaseFirestore.instance.collection('users');

  Future<void> setUser(UserModel user) async {
    try {
      _userReference.doc(user.id).set({
        'email': user.email,
        'name': user.name,
        'gender': user.gender,
        'date_of_birth': user.dateOfBirth,
      });
    } catch (e) {
      print(e);
    }
  }
}

```

```

    });
} catch (e) {
    throw Exception(e);
}
}

Future<UserModel> getUserById(String id) async {
try {
    DocumentSnapshot snapshot = await
_userReference.doc(id).get();
    return UserModel(
        id: id,
        email: snapshot['email'],
        name: snapshot['name'],
        gender: snapshot['gender'],
        dateOfBirth: snapshot['date_of_birth'],
    );
} catch (e) {
    throw Exception(e);
}
}

Future<UserModel> updateUser(
{required String label,
required String value,
required String userId}) async {
try {
    _userReference.doc(userId).update({
        label: value,
    });
    UserModel user = await getUserById(userId);
    return user;
} catch (e) {
    throw Exception(e);
}
}
}

```

43. auth_bloc.dart

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';

```

```
import 'package:healio/services/user_service.dart';

import '../../models/user_model.dart';
import '../../services/auth_service.dart';

part 'auth_event.dart';
part 'auth_state.dart';

class AuthBloc extends Bloc<AuthEvent, AuthState> {
    AuthBloc() : super(AuthInitial()) {
        on<AuthEvent>((event, emit) async {
            if (event is AuthSignUp) {
                try {
                    emit(AuthLoading());

                    UserModel user = await AuthService().signUp(
                        email: event.email,
                        password: event.password,
                        name: event.name,
                        gender: event.gender,
                        dateOfBirth: event.dateOfBirth,
                    );

                    emit(AuthSuccess(user));
                } catch (e) {
                    emit(AuthFailed(e.toString()));
                }
            }
            if (event is AuthSignIn) {
                try {
                    emit(AuthLoading());

                    UserModel user = await AuthService().signIn(
                        email: event.email,
                        password: event.password,
                    );

                    emit(AuthSuccess(user));
                } catch (e) {
                    emit(AuthFailed(e.toString()));
                }
            }
            if (event is AuthSignOut) {
                try {
```

```

        emit(AuthLoading());

        await AuthService().signOut();

        emit(AuthInitial());
    } catch (e) {
        emit(AuthFailed(e.toString()));
    }
}

if (event is AuthGetCurrentUser) {
    try {
        emit(AuthLoading());

        UserModel user = await
UserService().getUserById(event.id);

        emit(AuthSuccess(user));
    } catch (e) {
        emit(AuthFailed(e.toString()));
    }
}
}
}

```

44. auth_event.dart

```

part of 'auth_bloc.dart';

abstract class AuthEvent extends Equatable {
    const AuthEvent();

    @override
    List<Object> get props => [];
}

class AuthSignUp extends AuthEvent {
    final String email;
    final String password;
    final String name;
    final String gender;
}

```

```

final String dateOfBirth;
const AuthSignUp(
    this.email,    this.password,    this.name,    this.gender,
this.dateOfBirth);

@Override
List<Object> get props => [email, password, name, gender,
dateOfBirth];
}

class AuthSignIn extends AuthEvent {
final String email;
final String password;
const AuthSignIn(this.email, this.password);

@Override
List<Object> get props => [email, password];
}

class AuthSignOut extends AuthEvent {}

class AuthGetCurrentUser extends AuthEvent {
final String id;
const AuthGetCurrentUser(this.id);

@Override
List<Object> get props => [id];
}

```

45. auth_state.dart

```

part of 'auth_bloc.dart';

abstract class AuthState extends Equatable {
const AuthState();

@Override
List<Object> get props => [];
}

class AuthInitial extends AuthState {}

```

```

class AuthLoading extends AuthState {}

class AuthFailed extends AuthState {
    final String e;
    const AuthFailed(this.e);

    @override
    List<Object> get props => [e];
}

class AuthSuccess extends AuthState {
    final UserModel user;
    const AuthSuccess(this.user);

    @override
    List<Object> get props => [user];
}

```

46. health_bloc.dart

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:health/health.dart';

import '../../services/health_service.dart';

part 'health_event.dart';
part 'health_state.dart';

class HealthBloc extends Bloc<HealthEvent, HealthState> {
    HealthBloc() : super(HealthInitial()) {
        on<HealthEvent>((event, emit) async {
            if (event is HealthGetHeartRate) {
                try {
                    DateTime now = DateTime.now();
                    DateTime yesterday = now.subtract(const Duration(days: 1));
                    final types = [
                        HealthDataType.HEART_RATE,
                    ];
                    emit(HealthLoading());
                } catch (e) {
                    emit(HealthError(e));
                }
            }
        });
    }
}

```

```

        final heartRate =
            await HealthService().fetchData(now, yesterday,
types);

        emit(HealthHeartRateSuccess(heartRate));
    } catch (e) {
        emit(HealthFailed(e.toString()));
    }
}

if (event is HealthGetAllHeartRate) {
    try {
        DateTime now = DateTime.now();
        DateTime yesterday = now.subtract(const Duration(days:
30));
        final types = [
            HealthDataType.HEART_RATE,
        ];

        emit(HealthLoading());

        final allHeartRate =
            await HealthService().fetchData(now, yesterday,
types);

        emit(HealthAllHeartRateSuccess(allHeartRate));
    } catch (e) {
        emit(HealthFailed(e.toString()));
    }
}

if (event is HealthGetBloodOxygen) {
    try {
        DateTime now = DateTime.now();
        DateTime yesterday = now.subtract(const Duration(days:
1));
        final types = [
            HealthDataType.BLOOD_OXYGEN,
        ];

        emit(HealthLoading());

        final bloodOxygen =
            await HealthService().fetchData(now, yesterday,

```

```

types);

        emit(HealthBloodOxygenSuccess(bloodOxygen));
    } catch (e) {
        emit(HealthFailed(e.toString()));
    }
}

if (event is HealthGetAllBloodOxygen) {
    try {
        DateTime now = DateTime.now();
        DateTime yesterday = now.subtract(const Duration(days:
30));
        final types = [
            HealthDataType.BLOOD_OXYGEN,
        ];

        emit(HealthLoading());

        final bloodOxygen =
            await HealthService().fetchData(now, yesterday,
types);

        emit(HealthAllBloodOxygenSuccess(bloodOxygen));
    } catch (e) {
        emit(HealthFailed(e.toString()));
    }
}

if (event is HealthGetLandingPage) {
    try {
        DateTime now = DateTime.now();
        DateTime yesterday = now.subtract(const Duration(days:
1));
        final heartRateTypes = [
            HealthDataType.HEART_RATE,
        ];
        final bloodOxygenTypes = [
            HealthDataType.BLOOD_OXYGEN,
        ];

        emit(HealthLoading());

        final heartRate =
            await HealthService().fetchData(now, yesterday,

```

```
heartRateTypes);
    final bloodOxygen =
        await HealthService().fetchData(now, yesterday,
bloodOxygenTypes);

        emit(HealthLandingPageSuccess(heartRate, bloodOxygen));
    } catch (e) {
        emit(HealthFailed(e.toString()));
    }
}
});
}
}
```

47. health_event.dart

```
part of 'health_bloc.dart';

abstract class HealthEvent extends Equatable {
    const HealthEvent();

    @override
    List<Object> get props => [];
}

class HealthGetHeartRate extends HealthEvent {}

class HealthGetAllHeartRate extends HealthEvent {}

class HealthGetBloodOxygen extends HealthEvent {}

class HealthGetAllBloodOxygen extends HealthEvent {}

class HealthGetLandingPage extends HealthEvent {}
```

48. health_state.dart

```
part of 'health_bloc.dart';

abstract class HealthState extends Equatable {
```

```

const HealthState();

@Override
List<Object> get props => [];
}

class HealthInitial extends HealthState {}

class HealthLoading extends HealthState {}

class HealthFailed extends HealthState {
  final String e;
  const HealthFailed(this.e);

  @override
  List<Object> get props => [e];
}

class HealthHeartRateSuccess extends HealthState {
  final List<HealthDataPoint> heartRate;
  const HealthHeartRateSuccess(this.heartRate);

  @override
  List<Object> get props => [heartRate];
}

class HealthAllHeartRateSuccess extends HealthState {
  final List<HealthDataPoint> allHeartRate;
  const HealthAllHeartRateSuccess(this.allHeartRate);

  @override
  List<Object> get props => [allHeartRate];
}

class HealthBloodOxygenSuccess extends HealthState {
  final List<HealthDataPoint> bloodOxygen;
  const HealthBloodOxygenSuccess(this.bloodOxygen);

  @override
  List<Object> get props => [bloodOxygen];
}

class HealthAllBloodOxygenSuccess extends HealthState {
  final List<HealthDataPoint> allBloodOxygen;
  const HealthAllBloodOxygenSuccess(this.allBloodOxygen);
}

```

```

    @override
    List<Object> get props => [allBloodOxygen];
}

class HealthLandingPageSuccess extends HealthState {
    final List<HealthDataPoint> heartRate;
    final List<HealthDataPoint> bloodOxygen;
    const HealthLandingPageSuccess(this.heartRate, this.bloodOxygen);

    @override
    List<Object> get props => [heartRate, bloodOxygen];
}

```

49. heart_disease_bloc.dart

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';
import 'package:heilio/models/heart_disease_form_model.dart';
import 'package:heilio/services/heart_disease_service.dart';

part 'heart_disease_event.dart';
part 'heart_disease_state.dart';

class HeartDiseaseBloc extends Bloc<HeartDiseaseEvent,
HeartDiseaseState> {
    HeartDiseaseBloc() : super(HeartDiseaseInitial()) {
        on<HeartDiseaseEvent>((event, emit) async {
            if (event is HeartDiseasePrediction) {
                try {
                    emit(HeartDiseaseLoading());
                    final res = await HeartDiseaseService().prediction(event.data);
                    emit(HeartDiseaseSuccess(res));
                } catch (e) {
                    emit(HeartDiseaseFailed(e.toString()));
                }
            }
        });
    }
}

```

```
}
```

50. heart_disease_event.dart

```
part of 'heart_disease_bloc.dart';

abstract class HeartDiseaseEvent extends Equatable {
  const HeartDiseaseEvent();

  @override
  List<Object> get props => [];
}

class HeartDiseasePrediction extends HeartDiseaseEvent {
  final HeartDiseaseFormModel data;

  const HeartDiseasePrediction(this.data);

  @override
  List<Object> get props => [data];
}
```

51. heart_disease_state.dart

```
part of 'heart_disease_bloc.dart';

abstract class HeartDiseaseState extends Equatable {
  const HeartDiseaseState();

  @override
  List<Object> get props => [];
}

class HeartDiseaseInitial extends HeartDiseaseState {}

class HeartDiseaseLoading extends HeartDiseaseState {}

class HeartDiseaseFailed extends HeartDiseaseState {
  final String e;
  const HeartDiseaseFailed(this.e);
```

```

    @override
    List<Object> get props => [e];
}

class HeartDiseaseSuccess extends HeartDiseaseState {
    final Map<String, dynamic> data;
    const HeartDiseaseSuccess(this.data);

    @override
    List<Object> get props => [data];
}

```

52. page_cubit.dart

```

import 'package:bloc/bloc.dart';

class PageCubit extends Cubit<int> {
    PageCubit() : super(0);

    void setPage(int newPage) {
        emit(newPage);
    }
}

```

53. user_bloc.dart

```

import 'package:bloc/bloc.dart';
import 'package:equatable/equatable.dart';

import '../../models/user_model.dart';
import '../../services/user_service.dart';

part 'user_event.dart';
part 'user_state.dart';

class UserBloc extends Bloc<UserEvent, UserState> {
    UserBloc() : super(UserInitial()) {
        on<UserEvent>((event, emit) async {
            if (event is UserUpdate) {

```

```

    try {
        emit(UserLoading());

        UserModel user = await UserService().updateUser(
            label: event.label, value: event.value, userId:
        event.userId);

        emit(UserSuccess(user));
    } catch (e) {
        emit(UserFailed(e.toString()));
    }
}
);
}
}

```

54. user_event.dart

```

part of 'user_bloc.dart';

abstract class UserEvent extends Equatable {
    const UserEvent();

    @override
    List<Object> get props => [];
}

class UserUpdate extends UserEvent {
    final String label;
    final String value;
    final String userId;

    const UserUpdate(this.label, this.value, this.userId);

    @override
    List<Object> get props => [label, value, userId];
}

```

55. user_state.dart

```
part of 'user_bloc.dart';

abstract class UserState extends Equatable {
  const UserState();

  @override
  List<Object> get props => [];
}

class UserInitial extends UserState {}

class UserLoading extends UserState {}

class UserFailed extends UserState {
  final String e;
  const UserFailed(this.e);

  @override
  List<Object> get props => [e];
}

class UserSuccess extends UserState {
  final UserModel user;
  const UserSuccess(this.user);

  @override
  List<Object> get props => [user];
}
```