

LAMPIRAN

Lampiran 1. Surat Kesepakatan Bimbingan Dosen Pembimbing 1

SURAT KESEPAKATAN BIMBINGAN TUGAS AKHIR

Kami yang bertanda tangan di bawah ini :

Pihak Pertama

Nama : Angela Virgin Marcella
NIM : 22090017
Program Studi : Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama : Dyah Apriliani, S.T., M.Kom
Status : Dosen
NIDN : 0614049002
Jabatan Fungsional : Lektor
Pangkat/Golongan : III / D

Pada hari ini Rabu tanggal 11 Maret 2026 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing 1 Skripsi Pihak Pertama dengan syarat adanya peningkatan progres dari proyek skripsi yang dilaporkan setiap satu pekan. Jika Pihak Pertama tidak memenuhi syarat tersebut, maka Pihak Kedua berhak memutuskan kesepakatan ini. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi.

Tegal, 11 Maret 2026

Pihak Pertama


Angela Virgin Marcella

Pihak Kedua


Dyah Apriliani, S.T., M.Kom.
NIPY. 09.015.225

Mengetahui
Ketua Program Studi Sarjana Terapan
Teknik Informatika


Dyah Apriliani, S.T., M.Kom.
NIPY. 09.015.225

Lampiran 2. Surat Kesepakatan Bimbingan Dosen Pembimbing 2

SURAT KESEPAKATAN BIMBINGAN TUGAS AKHIR

Kami yang bertanda tangan di bawah ini:

Pihak Pertama

Nama : Angela Virgin Marcella
NIM : 22090017
Program Studi : Sarjana Terapan Teknik Informatika

Pihak Kedua

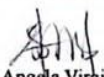
Nama : Sharfina Febbi Handayani, M.Kom.
Status : Dosen
NIDN : 0603088305
Jabatan Fungsional : Lektor
Pangkat/Golongan : Penata muda Tk. I/IIIb

Bersedia bimbingan secara rutin minimal 8 kali untuk menyelesaikan Skripsi, apabila dalam kurun waktu 3 bulan setelah kesepakatan ini dibuat pihak pertama tidak konsisten untuk menyanggupi maka pihak kedua berhak membatalkan kesepakatan ini dan pihak kedua tidak memberikan rekomendasi kepada pihak pertama untuk mendaftar sidang ujian skripsi.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi.

Tegal, 11 Maret 2025

Pihak Pertama


Angela Virgin Marcella

Pihak Kedua


Sharfina Febbi Handayani, M.Kom.
NIPY. 08.020.451

Mengetahui
Ketua Program Studi Sarjana Terapan
Teknik Informatika

Universitas Negeri Tegal
Dyah Abellum, S.T., M.Kom.
NIPY: 09.015.225

Lampiran 3. Surat Pernyataan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. **N a m a** : Angela Virgin Marcella
Kewarganegaraan : Indonesia
Alamat : Jl. Siuntung, Pesurungan Kidul, Kota Tegal
2. **N a m a** : Dyah Apriliani, S.T., M.Kom.
Kewarganegaraan : Indonesia
Alamat : Perumahan Mutiara Indah Blok C4, Jalan Nangka Gang 2, RT 02 RW 02, Kelurahan Procot Kec. Slawi, Kab. Tegal, Provinsi Jawa Tengah
3. **N a m a** : Sharfina Febbi Handayani, S.Kom., M.Kom.
Kewarganegaraan : Indonesia
Alamat : Grobog Kulon RT/RW 02/06 No. 46, Kecamatan Pangkah, Kabupaten Tegal, Provinsi Jawa Tengah 52471

Dengan ini menyatakan bahwa:

1. **Karya Cipta yang saya mohonkan:**
Berupa : Program Komputer
Berjudul : Pengembangan Aplikasi Mobile Stroke Shield untuk Deteksi Dini Risiko Stroke dan Rekomendasi Senam Pencegahan dengan Deteksi Gerakan Berbasis CNN
 - Tidak meniru dan tidak sama secara esensial dengan Karya Cipta milik pihak lain atau obyek kekayaan intelektual lainnya sebagaimana dimaksud dalam Pasal 68 ayat (2);
 - Bukan merupakan Ekspresi Budaya Tradisional sebagaimana dimaksud dalam Pasal 38;
 - Bukan merupakan Ciptaan yang tidak diketahui penciptanya sebagaimana dimaksud dalam Pasal 39;
 - Bukan merupakan hasil karya yang tidak dilindungi Hak Cipta sebagaimana dimaksud dalam Pasal 41 dan 42;
 - Bukan merupakan Ciptaan seni lukis yang berupa logo atau tanda pembeda yang digunakan sebagai merek dalam perdagangan barang/jasa atau digunakan sebagai lambang organisasi, badan usaha, atau badan hukum sebagaimana dimaksud dalam Pasal 65 dan;
 - Bukan merupakan Ciptaan yang melanggar norma agama, norma susila, ketertiban umum, pertahanan dan keamanan negara atau melanggar peraturan perundang-undangan sebagaimana dimaksud dalam Pasal 74 ayat (1) huruf d Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

2. Sebagai pemohon mempunyai kewajiban untuk menyimpan asli contoh ciptaan yang dimohonkan dan harus memberikan apabila dibutuhkan untuk kepentingan penyelesaian sengketa perdata maupun pidana sesuai dengan ketentuan perundang-undangan.
3. Karya Cipta yang saya mohonkan pada Angka 1 tersebut di atas tidak pernah dan tidak sedang dalam sengketa pidana dan/atau perdata di Pengadilan.
4. Dalam hal ketentuan sebagaimana dimaksud dalam Angka 1 dan Angka 3 tersebut di atas saya / kami langgar, maka saya / kami bersedia secara sukarela bahwa:
 - a. permohonan karya cipta yang saya ajukan dianggap ditarik kembali; atau
 - b. Karya Cipta yang telah terdaftar dalam Daftar Umum Ciptaan Direktorat Hak Cipta, Direktorat Jenderal Hak Kekayaan Intelektual, Kementerian Hukum Dan Hak Asasi Manusia R.I dihapuskan sesuai dengan ketentuan perundang-undangan yang berlaku.
 - c. Dalam hal kepemilikan Hak Cipta yang dimohonkan secara elektronik sedang dalam ber perkara dan/atau sedang dalam gugatan di Pengadilan maka status kepemilikan surat pencatatan elektronik tersebut ditangguhkan menunggu putusan Pengadilan yang berkekuatan hukum tetap.

Demikian Surat pernyataan ini saya/kami buat dengan sebenarnya dan untuk dipergunakan sebagaimana mestinya.

Tegal, 22 Juni 2026


Angela Virgin Marcella
Pemegang Hak Cipta *


Dyah Apriliani, S.T., M.Kom
Pemegang Hak Cipta *


Sharfina Febbi Handayani, S.Kom., M.Kom
Pemegang Hak Cipta *

* Semua pemegang hak cipta agar menandatangani di atas materai (untuk penulis pertama)

Lampiran 4. Surat Permohonan Izin Observasi



Universitas
Harkat Negeri

Fakultas Sekolah Vokasi
Program Studi D-4 Teknik Informatika

Nomor : 85.03/TI.HN/IV/2026
Lampiran : -
Hal : Permohonan Ijin Observasi
Kepada : Kepala RSUD Kardinah
Yth. Di Tempat

Dengan hormat, mahasiswa dengan identitas berikut ini:

nama : angela virgin marcella
NIM : 22090017
no hp : 081809624930
prodi : Sarjana Terapan Teknik Informatika

Bermaksud melakukan penelitian untuk keperluan Skripsi dengan judul **"pengembangan aplikasi stroke shield untuk deteksi dini risiko stroke dan rekomendasi senam pencegahan dengan deteksi gerakan berbasis CNN"**. Kami memohon Bapak/Ibu memberikan izin kepada mahasiswa yang bersangkutan agar dapat memperoleh data, keterangan, dan bahan yang diperlukan.

Demikian permohonan ini disampaikan, atas perhatian kami ucapkan terima kasih.

Tegal, 7 April 2026
Ka Prodi S.Tr. Teknik Informatika,


Universitas
Harkat Negeri
Dean Apri Illahi, S.T., M.Kom
MPY : 091015.225

Lampiran 5. Surat Izin Wawancara dan Validasi di Rehab Medik RS. Kardinah Tegal

INSTALASI DIKLAT

Hal : Ijin Pengambilan Data Awal

Tegal, 16 Maret 2026

Kepada Yth. :

Ka. Instalasi Rehab Medik

Dengan hormat,

Menindak lanjuti disposisi Direktur RSUD Kardinah pada surat dari

Nama Institusi : Universitas Harkat Negeri
Nomor Surat : 57.03/TI.HN/III/2026
Perihal : Permohonan Ijin Observasi
bersama ini kami mohon dapat diperkenankan kepada :
NAMA : ANGELA VIRGIN MARCELLA
Program Studi : D-IV Teknik Informatika

untuk mendapatkan data sesuai dengan keperluan dalam rangka penyusunan
KTI/Skripsi/tesis dengan Judul :

***"PENGEMBANGAN APLIKASI MOBILE STROKE SHIELD UNTUK EDUKASI
KESEHATAN DENGAN INTEGRASI CHATBOT BERBASIS NLP"***

Adapun persyaratan kelengkapan administrasi di RSUD Kardinah sudah dipenuhi.
Demikian untuk menjadikan maklum dan atas kerjasamanya terima kasih.

Ka. Sub. Bagian Kepegawaian & Pengembangan SDM
RSUD Kardinah Kota Tegal


DESI ERINI, A.Md.
PENATA
NIP. 19821220 200604 2 011

Lampiran 6. Surat Izin Wawancara dan Validasi Perawat di RS. Kardinah Tegal

INSTALASI DIKLAT

Hal : Ijin Pengambilan Data Awal

Tegal, 16 Maret 2026

Kepada Yth. :

Ka. Ruang Cendana 1

Dengan hormat,

Menindak lanjuti disposisi Direktur RSUD Kardinah pada surat dari

Nama Institusi : Universitas Harkat Negeri
Nomor Surat : 57.03/TI.HN/III/2026
Perihal : Permohonan Ijin Observasi
bersama ini kami mohon dapat diperkenankan kepada :
NAMA : ANGELA VIRGIN MARCELLA
Program Studi : D-IV Teknik Informatika

untuk mendapatkan data sesuai dengan keperluan dalam rangka penyusunan
KTI/Skripsi/tesis dengan Judul :

***"PENGEMBANGAN APLIKASI MOBILE STROKE SHIELD UNTUK EDUKASI
KESEHATAN DENGAN INTEGRASI CHATBOT BERBASIS NLP"***

Adapun persyaratan kelengkapan administrasi di RSUD Kardinah sudah dipenuhi.

Demikian untuk menjadikan maklum dan atas kerjasamanya terima kasih.

Ka. Sub. Bagian Kepegawaian & Pengembangan SDM
RSUD Kardinah Kota Tegal



Lampiran 7. Lembar bimbingan dosen pembimbing 1

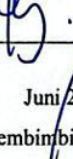


Sarjana Terapan Teknik Informatika
Universitas Harkat Negeri

LEMBAR BIMBINGAN SKRIPSI

Nama : Angela Virgin Marcella
NIM : 22090017
No. Ponsel : 081809624930
Judul Skripsi : Pengembangan aplikasi stroke shield untuk deteksi dini risiko stroke dan rekomendasi senam pencegahan dengan deteksi gerakan berbasis CNN
Dosen Pembimbing I : Dyah Apriliani, S.T., M.Kom

No.	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1.	11/3 2026	Judul.		
2.	22/4 2026	Aplikasi	- Data preuji. - Data set data set akan. - Data prebu. - Data. - Data set -> dokumen. - dokumen. - from lain. - alternatif.	
3.	6/4 2026	Observasi Validasi hasil screening		

4.	29/06/26	Progres aplikasi mobile	"tambahkan grafik di histori"	
5.	8/6/2026	Aplikasi	- Alter & Rupa bulan reban - Validasi (1) - admin (2) - fhm & - validasi (3) - Data HCL - form 1	
6.	24/06/26	Aplikasi	- Data HCL - form 1	
7.	26/06/2026	Laporan	- Revisi	
8.		laporan	- Revisi	
9.			Acc	

Tegal, Juni 2026
 Dosen Pembimbing

Dyah Apriliani, S.T., M.Kom.
 NIPY. 09.015.225

Lampiran 8. Lembar bimbingan dosen pembimbing 2



Sarjana Terapan Teknik Informatika
Universitas Harkat Negeri

LEMBAR BIMBINGAN SKRIPSI

Nama : Angela Virgin Marcella
NIM : 22090017
No. Ponsel : 081809624930
Judul Skripsi : Pengembangan aplikasi stroke shield untuk deteksi dini risiko stroke dan rekomendasi senam pencegahan dengan deteksi gerakan berbasis CNN
Dosen Pembimbing II : Sharfina Febbi Handayani, M.Kom

No.	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1.	11/3/2026	Judul	disesuaikan lagi dengan kebutuhan penderita stroke	Sharfi
2.	13/4/2026	Surat Kesepakatan membuat kuesioner	hasil kuesioner perlu divalidasi oleh ahli/pekar	Sharfi
3.	20/4/2026	alur apik	membuat PRD	
4.	06/05/2026	PRD role admin role user	Cek kembali gerakan senam berdasarkan jenis rendah, sedang, tinggi	Sharfi
5.	14/05/2026	Progres Aplikasi Mobile	Mencari data set video untuk gerakan di beberapa sosial media	
6.	23/05/2026	Progres Aplikasi Mobile	» diperbaiki kembali bagian login dengan google → screening » riwayat untuk gerakan baru atau riwayat screening » visualisasi grafik dg kolesterol dan gula darah	Sharfi
7.	03/06/2026	Progres Aplikasi Mobile	» dashboard sudah menampilkan grafik	Sharfi

8.	17/06/2026	Progres Aplikasi » bagian gerakan dan Informasi » Bagian dashboard admin	» perbaiki kembali menu informasi dan menu makan pada dashboard admin	Sharfi
9.	22/06/2026	Manual book Aplikasi Stroke Shield	» Perbaiki bagian deskripsi HKI pada menu Screening	Sharfi.
10	24/06/2026	Laporan Bab 1	Perbaiki bagian Latar belakang dan Tujuan Pustaka	Sharfi
11	26/06/2026	Laporan Bab 2	Perbaiki bagian Pengujian dg Kesimpulan	Sharfi
	28/06/2026	Laporan Bab 3	Diagram Pengujian HKI mohon ditambatkan	Sharfi
	29/06/2026	ACC Laporan	Lanjut Daftar Sidang	Sharfi

Tegal, 29 Juni 2026

Dosen Pembimbing II

Sharfi

Sharfina Febbi Handayani, S.Kom., M.Kom.

NIPY. 08.020.451

Lampiran 9. Wawancara terkait kategori risiko stroke



Lampiran 10. Validasi gerakan senam dengan perawat rehab medik



Lampiran 11. Validasi aplikasi



Lampiran 12. Manual book



Daftar Isi.....	1
Daftar Gambar.....	2
Pendahuluan	3
Tampilan Awal.....	4
Penggunaan Aplikasi.....	L-6
1. Registrasi dan Login	L-6
2. Dashboard	L-8
3. <i>Screening</i>	6
4. Hisori / Riwayat.....	L-12
5. Senam.....	L-13
6. Aktivitas.....	L-16
7. Rekomendasi Makanan.....	L-17
8. Info.....	L-18
9. Profile.....	L-19
10. Registrasi Admin.....	L-21
11. Login Admin	L-21
12. Dashboard Admin	L-22
13. Input Data Makanan.....	L-23
14. Daftar Menu	L-24
15. Profile.....	L-25
Pemecahan Masalah	L-27
1. Tidak Bisa Login.....	L-27
2. Aplikasi Tidak Merespon.....	L-27

DAFTAR GAMBAR

Gambar 1. Logo Stroke Shield.....	L-4
Gambar 2. Splas Screen	L-5
Gambar 3. Registrasi	L-6
Gambar 4. Login	L-8
Gambar 5. Dashboard.....	L-9
Gambar 6. <i>Screening</i>	L-11
Gambar 7. Histori.....	L-12
Gambar 8. Gerakan Senam	L-15
Gambar 9. Aktivitas	L-17
Gambar 10. Makanan	L-18
Gambar 11. Info	L-19
Gambar 12. Profile	L-20
Gambar 13. Registrasi Admin.....	L-21
Gambar 14. Login Admin	L-22
Gambar 15. Dashboard Admin	L-23
Gambar 16. Input Data Makanan	L-24
Gambar 17. Daftar Menu	L-25
Gambar 18. Profile Admin	L-26

1. PENDAHULUAN

Selamat datang di Stroke Shield, sebuah aplikasi edukasi kesehatan dan skrining mandiri yang dirancang khusus untuk membantu Anda menjaga kesehatan tubuh dan mencegah risiko penyakit stroke dengan lebih baik. Dalam era modern di mana gaya hidup yang kurang aktif dan pola makan tidak seimbang kerap menjadi tantangan, pencegahan penyakit kritis seperti stroke menjadi aspek yang sangat penting untuk diperhatikan. Stroke Shield hadir untuk memberikan dukungan yang Anda butuhkan melalui berbagai fitur yang berfokus pada tindakan preventif, mulai dari pemantauan risiko kesehatan, panduan senam pencegahan, hingga penyediaan artikel edukatif.

Stroke Shield dirancang dengan mempertimbangkan kebutuhan pengguna dari berbagai latar belakang dan kelompok usia. Aplikasi ini menawarkan pendekatan yang preventif dan personal dalam upaya mendeteksi serta menurunkan faktor risiko stroke. Dengan menggunakan teknologi canggih berbasis kecerdasan buatan (*Artificial Intelligence*), Stroke Shield berkomitmen untuk menyediakan layanan pemantauan kesehatan mandiri yang berkualitas tinggi, informatif, dan mudah diakses oleh siapa saja, kapan saja.



Gambar 1. Logo Stroke Shield

Saat Anda pertama kali membuka aplikasi Stroke Shield, Anda akan melihat splash screen seperti yang ditunjukkan pada gambar berikut:



Gambar 2. Splas Screen

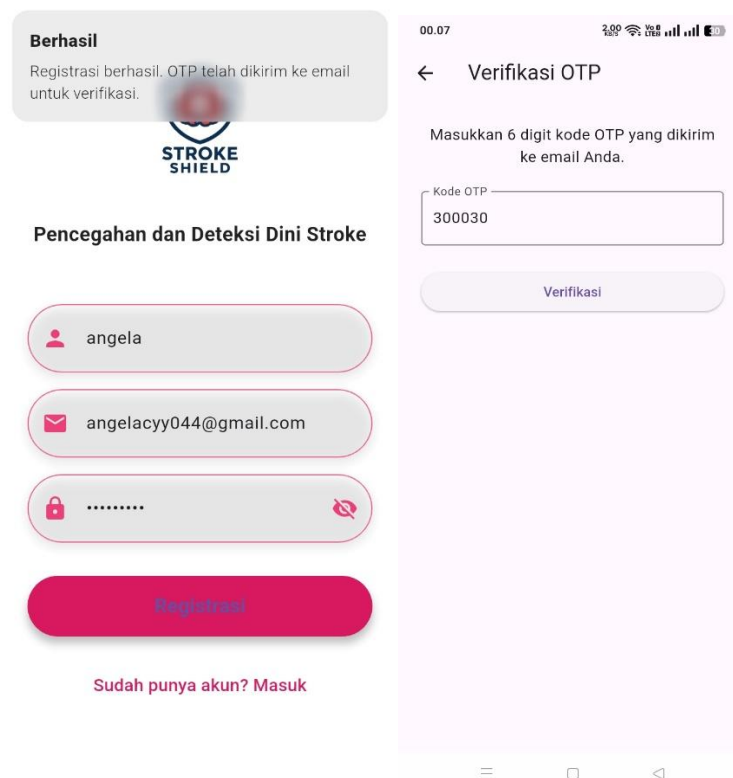
Pada *splash screen* ini, Anda akan melihat logo resmi aplikasi yang diikuti dengan nama "Stroke Shield". Di bawah nama aplikasi, terdapat deskripsi singkat yang menjelaskan bahwa aplikasi ini dirancang untuk membantu pencegahan dan deteksi dini stroke melalui edukasi serta pemantauan risiko kesehatan. Selain itu, halaman ini juga menyediakan dua tombol utama yang memungkinkan pengguna untuk masuk ke dalam aplikasi ("Login") atau mendaftarkan akun baru ("Sign Up"). Setelah pengguna memilih untuk masuk atau mendaftar, aplikasi akan diarahkan ke halaman utama (Beranda) yang menampilkan sapaan personal untuk pengguna beserta beragam fitur yang tersedia

Penggunaan Aplikasi

2. Registrasi dan Login

a. Registrasi

- 1) Buka aplikasi Stroke Shield.
- 2) Pilih "Registrasi" untuk membuat akun baru.
- 3) Isi formulir pendaftaran dengan informasi yang diminta, seperti, email, nama dan password.
- 4) Klik "Registrasi" untuk menyelesaikan proses pendaftaran.
- 5) Kemudian masukan Kode OTP yang terkirim ke email yang sudah di daftarkan.



Gambar 3. Registrasi

b. Login

- 1) Buka aplikasi Stroke Shield.
- 2) Pilih "Login".
- 3) Masukkan email dan kata sandi yang telah Anda daftarkan.
- 4) Klik "Login" untuk masuk ke akun Anda.

00.07

0.21 KB/s VoLTE



Pencegahan dan Deteksi Dini Stroke

Selamat Datang

Masuk untuk melanjutkan
dan pantau risiko stroke Anda

 angelacyy044@gmail.com





[Lupa Kata Sandi?](#)

Masuk

 Masuk dengan Google

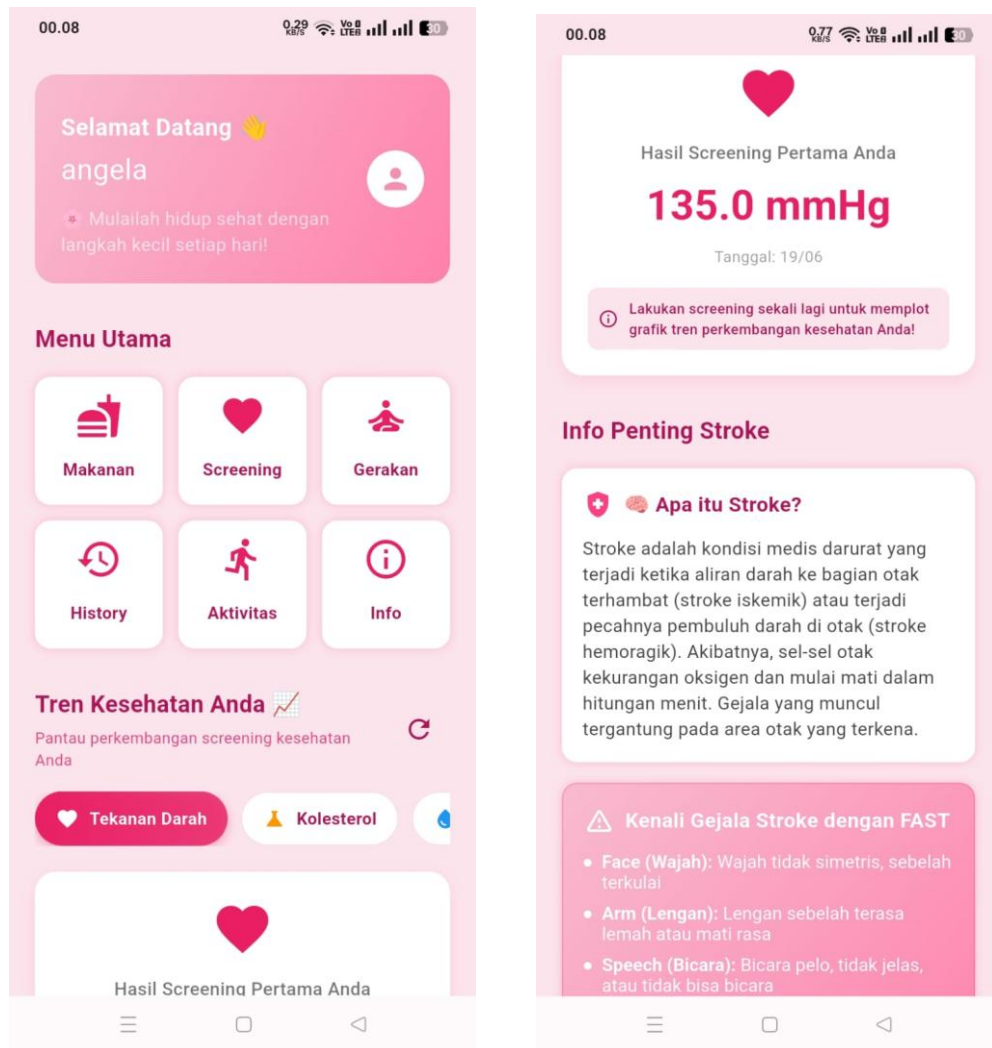
Belum punya akun? [Daftar](#)



Gambar 4. Login

3. Dashboard

Setelah berhasil login, Anda akan diarahkan ke dashboard. Di sini Anda dapat mengakses berbagai fitur utama aplikasi, yaitu Makanan, *Screening*, Gerakan, Histori, Aktivitas, Info, Profile. Selain itu, *dashboard* juga menyajikan visualisasi grafik dari hasil *Screening* Anda, serta sekilas informasi edukatif mengenai pengertian dan gejala penyakit stroke.



Gambar 5. Dashboard

4. *Screening*

Aplikasi ini menyediakan fitur deteksi dini (*Screening*) risiko stroke dengan serangkaian pertanyaan terkait kondisi kesehatan dan gaya hidup yang dirancang khusus untuk membantu Anda mengevaluasi tingkat risiko stroke Anda. Berikut cara mengaksesnya:

- Pilih menu "*Screening*" di halaman utama (Beranda).

b. Jawab setiap pertanyaan terkait metrik kesehatan Anda (seperti umur, jenis kelamin, berat badan, tinggi badan , BMI (otomatis), tekanan darah, kolesterol, riwayat keluarga, dan aktivitas fisik) secara jujur dan akurat. Berikut adalah penjelasan mengenai metrik pertanyaan yang akan Anda temui pada saat *Screening*:

- Umur: Masukkan usia Anda saat ini, karena risiko stroke umumnya akan meningkat seiring bertambahnya usia.
- Jenis Kelamin: Pilih Pria atau Wanita untuk menyesuaikan profil risiko yang berbeda pada setiap gender.
- Berat Badan: Anda diminta untuk memasukkan berat badan saat ini (dalam satuan kg) sebagai data dasar pengukuran fisik.
- Tinggi Badan: Anda diminta untuk memasukkan tinggi badan saat ini (dalam satuan cm).
- BMI (*Body Mass Index*): Angka proporsi tubuh yang dihitung otomatis oleh sistem untuk mendeteksi potensi *overweight* atau obesitas.
- Riwayat Keluarga: Tentukan apakah ada keluarga inti Anda yang memiliki riwayat stroke untuk melihat faktor genetika.
- Merokok: Pilih status kebiasaan merokok Anda, karena paparan asap rokok sangat memengaruhi kesehatan pembuluh darah.
- Pola Makan: Nilai kebiasaan makan Anda (misalnya: Sehat / Tidak Sehat) untuk mengevaluasi konsumsi gizi harian Anda.
- Aktivitas Fisik: Tentukan tingkat intensitas gerak tubuh atau rutinitas olahraga Anda sehari-hari.
- Tekanan Darah: Masukkan nilai tekanan darah (dalam mmHg) sebagai salah satu indikator risiko utama pemicu stroke.
- Kolesterol: Masukkan kadar kolesterol Anda untuk memantau potensi penyumbatan pada aliran darah.

- Gula Darah: Masukkan kadar gula darah Anda, karena kadar gula yang tinggi secara terus-menerus dapat merusak pembuluh darah.
- c. Setelah selesai menekan tombol prediksi, Anda akan langsung mendapatkan hasil evaluasi tingkat risiko stroke (Tinggi/Rendah).

The image displays two screenshots of a mobile application titled "Deteksi Dini Stroke".

Left Screenshot (00.08):

- Data Pribadi:**
 - Umur: Input field with a birthday icon.
 - Jenis Kelamin: Buttons for "Wanita" and "Pria".
 - Berat Badan (kg): Input field with a weight icon.
 - Tinggi Badan (cm): Input field with a height icon.
 - BMI: Input field with a BMI icon and an information icon.
- Gaya Hidup & Riwayat:**
 - Riwayat Keluarga: Buttons for "YA" and "TIDAK".

Right Screenshot (00.09):

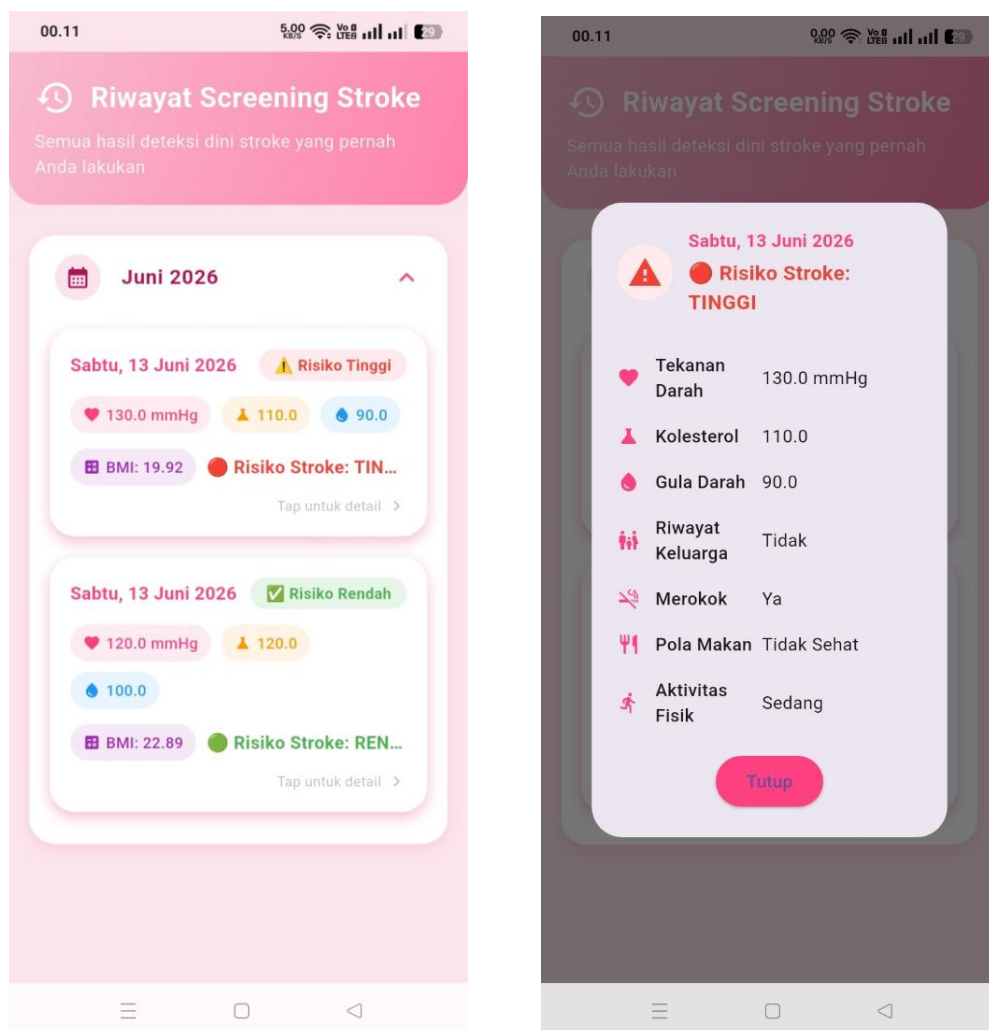
- Exercise Frequency:** Radio button options:
 - Hampir tidak pernah olahraga atau hanya sesekali
 - Sedang: Olahraga beberapa kali dalam seminggu
 - Sering: Olahraga hampir setiap hari
- Data Medis:**
 - Tekanan Darah: Input field with a heart icon.
 - Kolesterol: Input field with a cholesterol icon.
 - Gula Darah: Input field with a blood drop icon.
- Prediksi Risiko Stroke:** A prominent pink button at the bottom.

Gambar 6. Screening

5. Hisori / Riwayat

Aplikasi ini menyediakan fitur riwayat (*history*) yang memungkinkan Anda untuk menyimpan, melacak, dan memantau perkembangan hasil *Screening* risiko stroke Anda dari waktu ke waktu. Berikut cara mengaksesnya:

- Pilih menu "History" pada halaman utama (Beranda) aplikasi.
- Anda akan diarahkan ke halaman "Riwayat *Screening* Stroke" yang menampilkan daftar seluruh hasil deteksi dini yang pernah Anda lakukan, yang dikelompokkan berdasarkan bulan dan tanggal.
- Ketuk tulisan "Tap untuk detail >" pada salah satu riwayat untuk membuka jendela informasi (opsi *pop-up*). Jendela ini akan menampilkan rincian lengkap mengenai hasil prediksi akhir.



Gambar 7. Histori

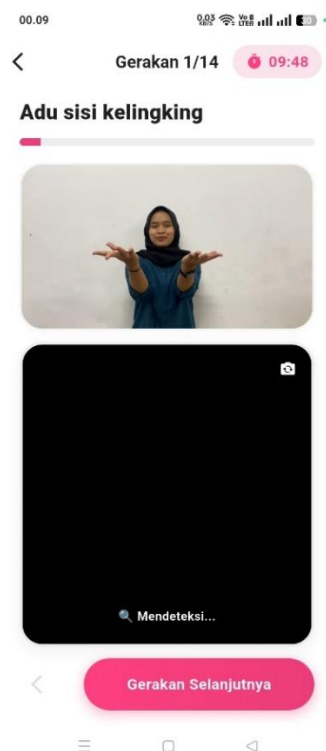
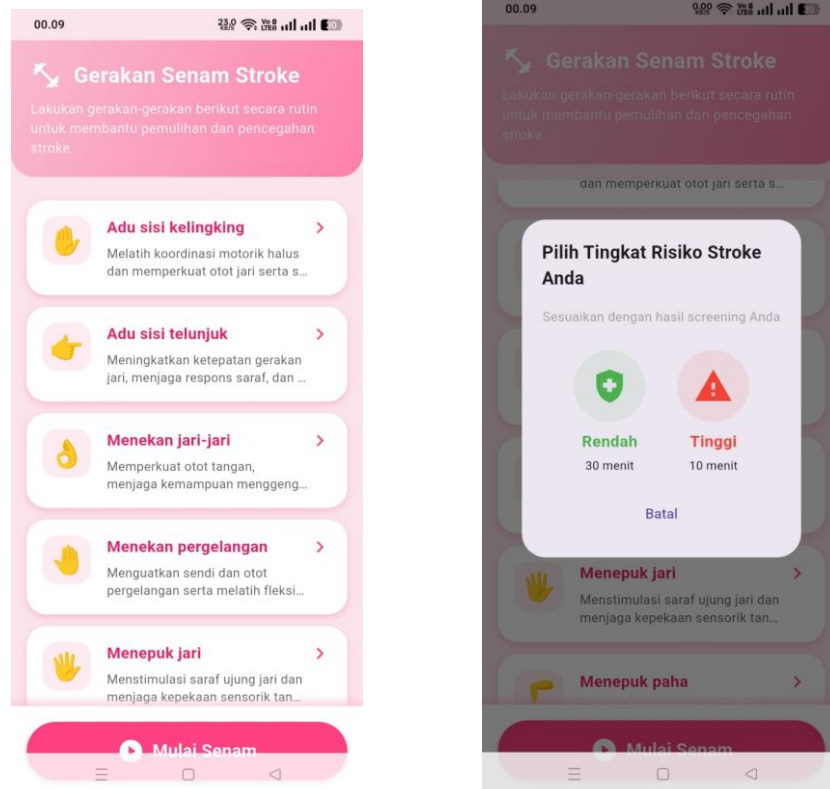
6. Senam

Gerakan Senam Stroke yang interaktif untuk membantu Anda melakukan aktivitas fisik secara rutin guna mendukung proses pemulihan dan pencegahan penyakit stroke. Berikut cara mengakses dan menggunakan fitur ini:

- a. Pilih menu "Gerakan" pada halaman utama (Beranda) aplikasi.
- b. Anda akan diarahkan ke halaman daftar "Gerakan Senam Stroke". Halaman ini menyajikan berbagai macam edukasi gerakan.
- c. Untuk memulai sesi latihan, tekan tombol "Mulai Senam" yang berada di bagian bawah layar.
- d. Sistem akan menampilkan *pop-up* yang meminta Anda untuk memilih tingkat risiko stroke Anda (Tinggi atau Rendah).
- e. Setelah durasi dipilih, Anda akan masuk ke mode senam interaktif. Pada halaman ini, bagian atas layar akan memutar video tutorial gerakan yang harus ditiru beserta batas waktu mundur (*timer*). Sementara itu, bagian bawah layar akan mengaktifkan kamera perangkat Anda secara *real-time* (Mendeteksi).
- f. Anda dapat menekan tombol "Gerakan Selanjutnya" untuk beralih ke instruksi latihan berikutnya hingga seluruh sesi senam selesai.
- g. Secara keseluruhan, fitur ini menyediakan **14 jenis gerakan** senam ringan yang dirancang khusus untuk melatih koordinasi motorik dan respons saraf. Berikut penjelasan setiap gerakan:

1. Adu Sisi Kelingking: melatih koordinasi motorik halus dan memperkuat otot jari serta stimulasi saraf sensorik
2. Adu Sisi Telunjuk: meningkatkan ketepatan gerakan jari, menjaga respons saraf, dan melatih konsentrasi
3. Menekan Jari-Jari: memperkuat otot tangan, menjaga kemampuan mengpegang, dan melancarkan peredaran darah.
4. Menekan Pergelangan: menguatkan sendi dan otot pergelangan serta melatih fleksibilitas tangan.
5. Menepuk Jari: menstimulasi saraf ujung jari dan menjaga kepekaan sensorik tangan.

6. Menepuk Paha: meningkatkan sirkulasi darah di paha dan melatih kepekaan tubuh bagian bawah.
7. Menepuk Perut: melatih kesadaran tubuh, memperkuat otot inti, dan merangsang sistem pencernaan.
8. Menepuk Pinggang: menjaga kelenturan bagian tengah dan mencegah kekakuan otot pinggang.
9. Menepuk Punggung Tangan: melancarkan aliran darah dan menjaga kepekaan saraf tangan.
10. Menepuk samping betis: merangsang otot betis dan memperbaiki sirkulasi darah ke kaki
11. Menepuk Sela Jari-Jari: merangsang saraf sensorik dan menjaga elastisitas kulit tangan
12. Mengetuk Lengan dan Bahu: Melemaskan otot yang tegang dan melancarkan aliran darah tubuh bagian atas
13. Silang Ibu Jari: meningkatkan konsentrasi, koordinasi bilateri, dan kerja otak kanan dan kiri.
14. Tepuk Tangan: meningkatkan kesadaran gerak, sirkulasi darah, dan memberikan efek relaksasi



Gambar 8. Gerakan Senam

7. Aktivitas

Sebagai bentuk evaluasi latihan fisik, aplikasi ini dilengkapi dengan fitur Aktivitas yang berfungsi untuk merekam dan melacak performa senam Anda secara terperinci. Berikut adalah informasi yang ditampilkan dan cara memahami

- a. Buka menu "Aktivitas" pada aplikasi.
- b. Halaman ini menampilkan daftar riwayat gerakan senam pencegahan stroke yang telah Anda selesaikan, yang dikelompokkan per bulan.
- c. Setiap riwayat memuat detail performa Anda, meliputi: waktu pelaksanaan, nama gerakan, tingkat akurasi, dan durasi gerakan yang berhasil dilakukan dengan benar.

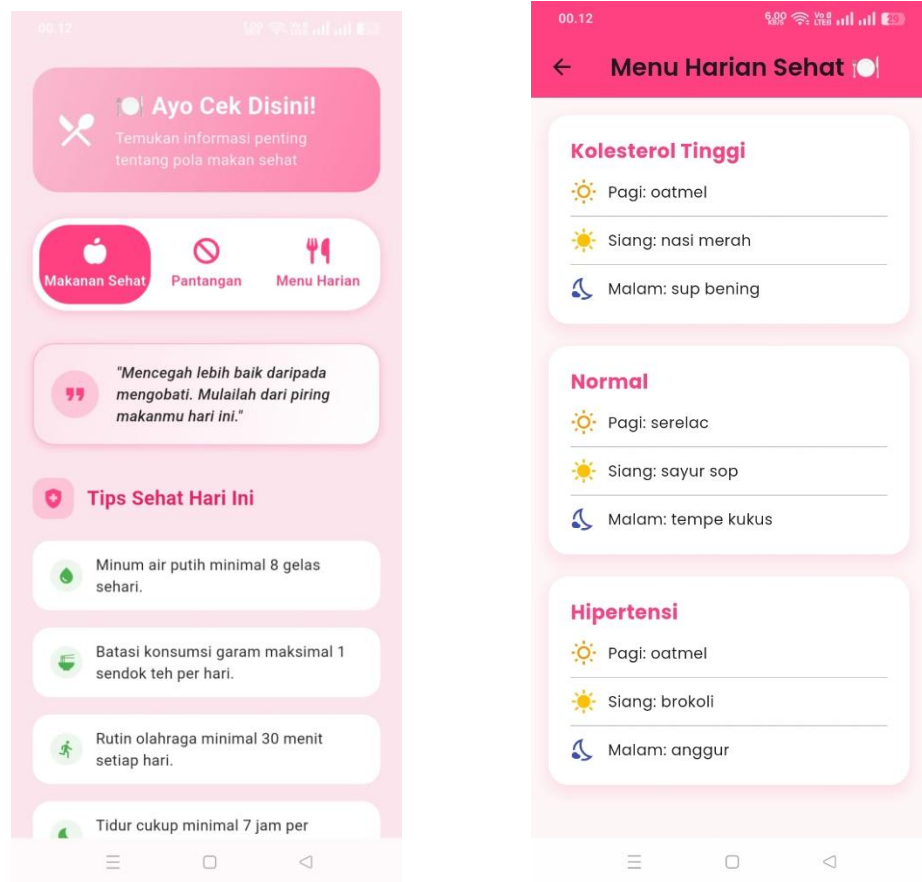


Gambar 9. Aktivitas

8. Rekomendasi Makanan

Sebagai pendukung gaya hidup sehat, aplikasi ini menawarkan fitur Makanan untuk membantu pengguna menjaga pola makan seimbang. Berikut panduan singkat penggunaannya:

- a. Buka menu "Makanan" pada halaman utama aplikasi.
- b. Halaman awal akan menyajikan edukasi seperti "Tips Sehat Hari Ini" beserta tiga pilihan utama: Makanan Sehat, Pantangan, dan Menu Harian.
- c. Pilih opsi "Menu Harian" untuk melihat rekomendasi menu makanan sehat setiap hari. Jadwal menu (pagi, siang, dan malam) ini disesuaikan secara khusus dengan kondisi kesehatan Anda, seperti Kolesterol Tinggi, Hipertensi, atau Normal.



Gambar 10. Makanan

9. Info

Untuk memperkaya wawasan dan literasi kesehatan Anda, aplikasi ini menyajikan fitur Info. Berikut panduan singkat penggunaannya:

- Pilih menu "Info" pada halaman utama (Beranda) aplikasi.
- Anda akan diarahkan ke halaman Informasi Kesehatan interaktif.
- Pada halaman ini, Anda dapat mengeksplorasi visualisasi data seputar gejala stroke, melihat statistik umum, serta mengakses daftar ratusan artikel kesehatan untuk memperdalam pemahaman Anda mengenai penyakit stroke.

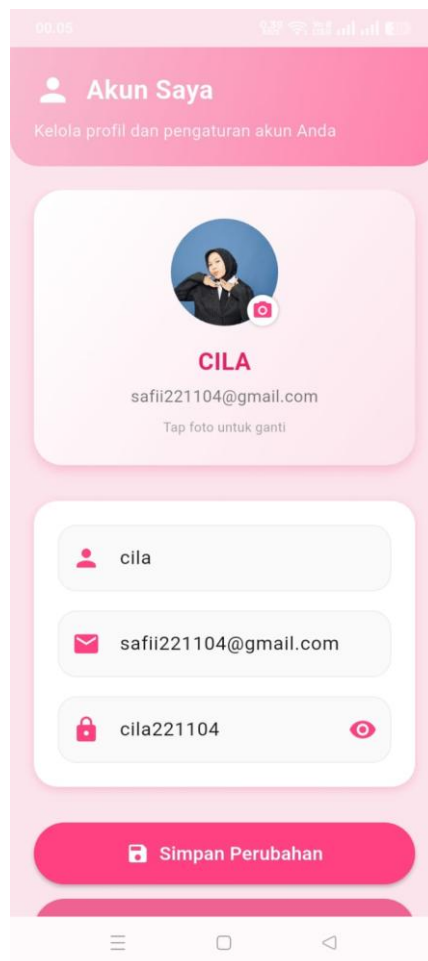


Gambar 11. Info

10. Profile

Untuk memfasilitasi pengelolaan data secara mandiri, aplikasi ini dilengkapi dengan fitur Profile. Berikut panduan singkat penggunaannya:

- a. Pada halaman utama (*Dashboard*), ketuk foto profil Anda yang terletak di samping teks sapaan "Selamat Datang". B.
- b. Pada halaman ini, Anda dapat memperbarui data diri, seperti mengubah foto profil, mengganti nama, atau memperbarui kata sandi.
- c. Setelah selesai melakukan penyesuaian, tekan tombol "**Simpan Perubahan**" agar data profil Anda yang baru diperbarui dan tersimpan di dalam sistem.

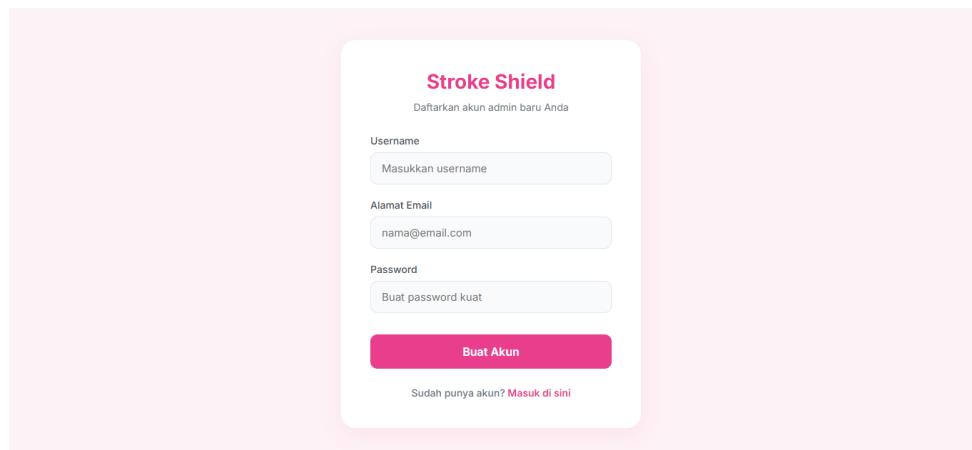


Gambar 12. Profile

11. Registrasi Admin

Sebelum dapat masuk dan mengelola sistem pada *dashboard* admin, Anda diwajibkan untuk membuat akun pengelola terlebih dahulu. Berikut panduan singkat pengisian formulir pendaftarannya:

- a. Masukkan Username (nama pengguna) yang akan Anda gunakan sebagai identitas profil di dalam sistem.
- b. Masukkan Alamat Email aktif yang valid. Email ini akan digunakan sebagai kredensial utama saat masuk (*login*) dan keperluan pemulihan akun.
- c. Buat Password (kata sandi) yang kuat untuk melindungi keamanan akses data admin Anda.
- d. Setelah seluruh kolom dipastikan terisi dengan benar, tekan tombol "Buat Akun" agar sistem memproses dan menyimpan pendaftaran Anda.
- e. Sebagai catatan tambahan, jika Anda sudah memiliki akun admin sebelumnya, Anda dapat langsung menekan tautan "Masuk di sini" di bagian bawah layar untuk beralih ke halaman *login*.

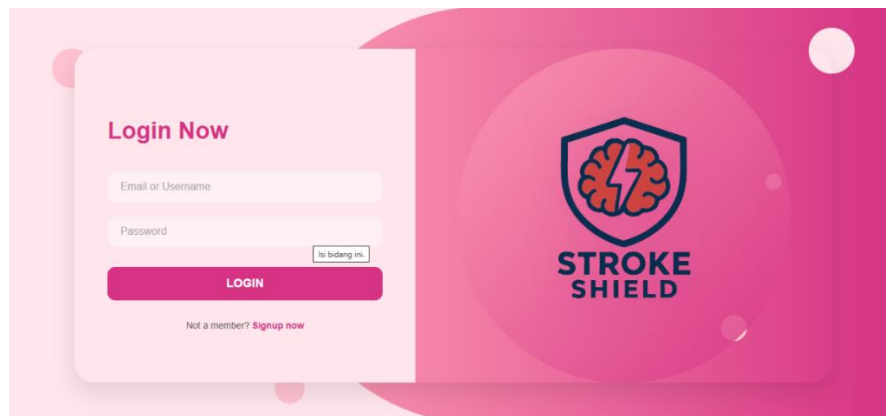
The image shows a registration form for 'Stroke Shield'. The form is titled 'Stroke Shield' in pink and 'Daftarkan akun admin baru Anda' below it. It contains three input fields: 'Username' with the placeholder 'Masukkan username', 'Alamat Email' with the placeholder 'nama@email.com', and 'Password' with the placeholder 'Buat password kuat'. Below these fields is a pink button labeled 'Buat Akun'. At the bottom, there is a link that says 'Sudah punya akun? Masuk di sini'.

Gambar 13. Registrasi Admin

12. Login Admin

Setelah memiliki akun, Anda dapat masuk ke dalam sistem untuk mulai mengelola aplikasi Stroke Shield. Berikut adalah panduan langkah-langkahnya:

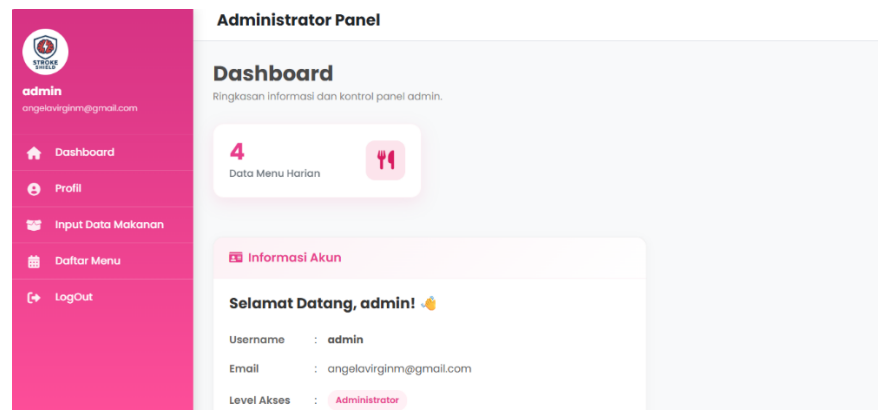
- a. Masukkan Email atau Username yang telah Anda daftarkan sebelumnya pada kolom teks pertama.
- b. Masukkan Password (kata sandi) Anda pada kolom kedua. Pastikan kata sandi diketik dengan tepat dan perhatikan penggunaan huruf besar atau kecil.
- c. Tekan tombol "login" yang berwarna merah muda. Jika data benar, sistem akan memverifikasi dan mengarahkan Anda ke *dashboard* utama admin.
- d. Sebagai tambahan, apabila Anda belum mendaftarkan akun sama sekali, Anda dapat mengklik tautan "Signup now" di bagian bawah untuk beralih ke halaman pendaftaran.



Gambar 14. Login Admin

13. Dashboard Admin

Setelah berhasil *login*, Anda akan diarahkan ke *dashboard Administrator Panel*. Di sini Anda dapat mengakses berbagai fitur pengelolaan utama aplikasi melalui menu di sisi kiri, yaitu Profil, Input Data Makanan, Daftar Menu, dan opsi LogOut. Selain itu, *dashboard* juga menyajikan ringkasan statistik ketersediaan Data Menu Harian, serta panel Informasi Akun yang menampilkan detail *Username*, *Email*, dan Level Akses Anda.



Gambar 15. Dashboard Admin

14. Input Data Makanan

Pada halaman ini, Anda dapat mengatur dan menambahkan jadwal asupan gizi harian yang disesuaikan dengan kondisi medis tertentu. Berikut adalah panduan pengisian formnya:

- Informasi Dasar: Tentukan Tanggal berlakunya jadwal menu tersebut, kemudian pilih jenis Kondisi Kesehatan melalui menu pilihan (*dropdown*) yang tersedia.
- Detail Menu Makan: Masukkan rincian asupan makanan secara lengkap dan jelas pada masing-masing kolom, yaitu Menu Pagi, Menu Siang, dan Menu Malam (contoh pengisian: Nasi merah, ayam rebus, brokoli).
- Setelah seluruh rincian menu dipastikan sesuai, tekan tombol "Simpan Data Menu" berwarna merah muda di bagian bawah untuk menyimpan jadwal asupan gizi tersebut ke dalam basis data sistem.

Administrator Panel

admin
angelavirgin@gmail.com

- Dashboard
- Profil
- Input Data Makanan
- Daftar Menu
- LogOut

Input Menu Harian Kesehatan

Atur jadwal asupan gizi harian sesuai dengan kondisi medis.

Informasi Dasar

Tanggal: Kondisi Kesehatan:

Detail Menu Makan

Menu Pagi
Contoh: Oatmeal, pisang, teh tanpa gula

Menu Siang
Contoh: Nasi merah, ayam rebus, brokoli

Menu Malam
Contoh: Sup bening, tahu kukus, pepaya

Simpan Data Menu

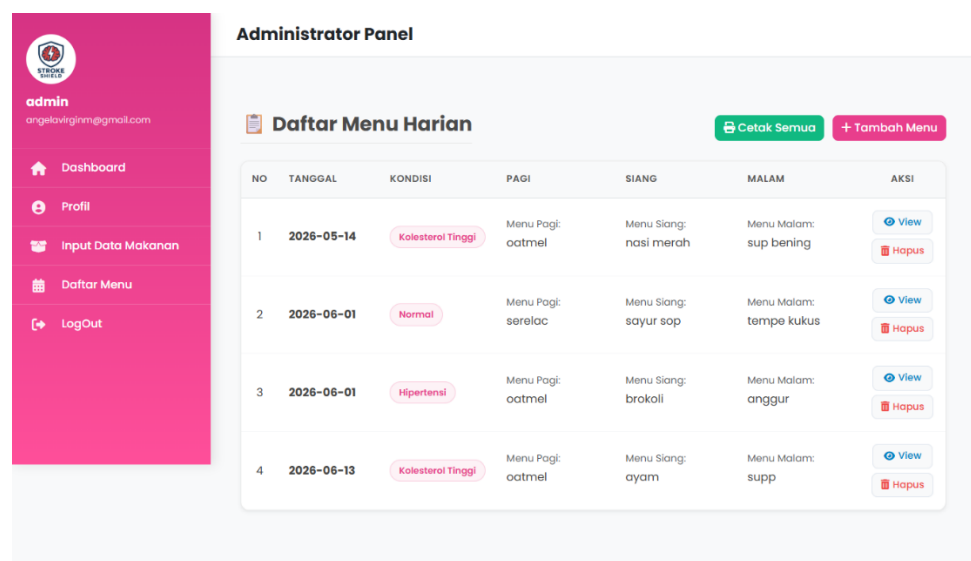
Gambar 16. Input Data Makanan

15. Daftar Menu

Halaman ini berfungsi sebagai pusat pengelolaan rekapitulasi data, di mana Anda dapat melihat seluruh riwayat jadwal asupan gizi yang telah Anda simpan ke dalam sistem. Berikut adalah panduan fungsi pada halaman ini:

- Tabel Daftar Menu Harian: Menampilkan rekap data secara terstruktur ke dalam beberapa kolom informasi, yang meliputi: Tanggal, status Kondisi medis pasien (seperti Normal, Hipertensi, atau Kolesterol Tinggi), detail ringkasan menu (Pagi, Siang, Malam), serta kolom Aksi.
- Tombol Aksi (Tabel): Pada kolom paling kanan di setiap baris data, Anda dapat menggunakan tombol "View" untuk melihat detail rincian menu secara lebih spesifik, atau menekan tombol "Hapus" (berwarna merah) untuk menghapus data menu tersebut dari sistem secara permanen.

- c. Cetak Semua: Anda dapat menekan tombol berwarna hijau ini yang terletak di pojok kanan atas apabila Anda ingin mencetak (*print*) seluruh rekap daftar menu harian menjadi sebuah dokumen.
- d. Tambah Menu: Tombol merah muda di sebelah kanan atas ini berfungsi sebagai jalan pintas (*shortcut*) untuk mengarahkan Anda kembali ke halaman *Input Data Makanan* apabila Anda ingin membuat jadwal asupan gizi yang baru.



Administrator Panel

admin
angelavirgin@gmail.com

- Dashboard
- Profil
- Input Data Makanan
- Daftar Menu
- Logout

Daftar Menu Harian

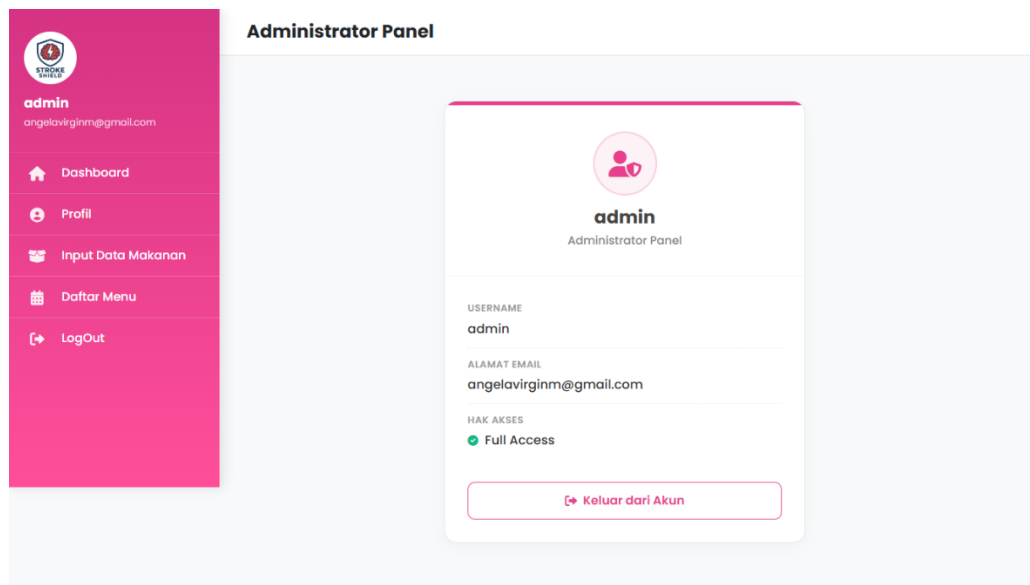
Cetak Semua + Tambah Menu

NO	TANGGAL	KONDISI	PAGI	SIANG	MALAM	AKSI
1	2026-05-14	Kolesterol Tinggi	Menu Pagi: oatmel	Menu Siang: nasi merah	Menu Malam: sup bening	View Hapus
2	2026-06-01	Normal	Menu Pagi: serealac	Menu Siang: sayur sop	Menu Malam: tempe kukus	View Hapus
3	2026-06-01	Hipertensi	Menu Pagi: oatmel	Menu Siang: brokoli	Menu Malam: anggur	View Hapus
4	2026-06-13	Kolesterol Tinggi	Menu Pagi: oatmel	Menu Siang: ayam	Menu Malam: supp	View Hapus

Gambar 17. Daftar Menu

16. Profile

Halaman Profil difungsikan untuk menampilkan detail informasi identitas dari akun admin yang sedang Anda gunakan. Pada halaman ini, Anda dapat meninjau rincian data akun yang meliputi *Username* dan Alamat Email yang terdaftar, serta melihat status Hak Akses (*Full Access*) yang menandakan bahwa Anda memiliki otoritas kendali penuh di dalam sistem. Selain itu, halaman ini juga dilengkapi dengan tombol "Keluar dari Akun" di bagian bawah yang dapat Anda tekan untuk mengakhiri sesi (*Logout*) secara aman setelah selesai mengelola *Administrator Panel*.



Gambar 18. Profile Admin

Pemecahan Masalah

Pemecahan Masalah

Tidak Bisa Login

- a. Periksa Koneksi Internet: Pastikan perangkat Anda terhubung dengan jaringan internet yang stabil. Stroke Shield memerlukan koneksi aktif untuk menyinkronkan data rekam medis dan *Screening* Anda.
- b. Cek Kredensial Akun: Pastikan alamat email dan kata sandi yang Anda masukkan sudah benar. Jika Anda menggunakan opsi pendaftaran praktis, pastikan akun Google yang tertaut masih aktif di perangkat Anda.

Aplikasi Tidak Merespon

- a. Mulai Ulang Aplikasi: Tutup aplikasi Stroke Shield sepenuhnya dari daftar aplikasi yang berjalan di latar belakang (*recent apps*), kemudian buka kembali.

Lampiran 13. Dokumen teknis



Daftar Isi.....	1
Pendahuluan	2
Spesifikasi Teknis	M-8
1. Import Library.....	M-9
2. Konfigurasi Database.....	M-11
3. Route Register.....	M-12
4. Route Login	M-14
5. Route Verify Otp	M-15
6. Route Edit Password.....	M-17
7. Route Login Google.....	M-19
8. Route <i>Screening</i> History	M-21
9. Route <i>Screening</i>	M-22
10. Route Senam	M-26

Selamat datang di Stroke Shield, sebuah aplikasi *mobile* edukasi kesehatan yang dirancang khusus untuk membantu Anda mencegah dan mendeteksi dini risiko penyakit stroke dengan lebih baik. Dalam era modern di mana akses informasi kesehatan sangat krusial, pencegahan penyakit kritis menjadi aspek yang sangat penting untuk diperhatikan. Stroke Shield hadir untuk memberikan dukungan yang Anda butuhkan melalui fitur pemantauan risiko kesehatan berbasis skrining (*Screening*) dan panduan senam pencegahan yang interaktif.

Latar belakang dari aplikasi ini adalah tingginya bahaya stroke sebagai salah satu penyakit penyebab kematian dan disabilitas utama secara global, yang menimbulkan beban kesehatan, sosial, dan ekonomi yang signifikan. Keadaan ini muncul saat aliran darah ke otak terhambat, baik akibat sumbatan atau robeknya pembuluh darah, yang menyebabkan sel-sel otak mati dalam waktu yang singkat. Angka kejadian stroke yang melambung tinggi menjadikan isu kesehatan masyarakat ini perlu segera diatasi, baik di negara maju maupun di negara berkembang seperti Indonesia.

Di Indonesia, penyakit stroke menunjukkan kecenderungan yang semakin memprihatinkan. Berdasarkan pada data Riset Kesehatan Dasar (Riskesmas), prevalensi stroke meningkat 7% pada tahun 2013 dan menjadi 10,9% pada tahun 2018. Peningkatan ini mencerminkan bahwa jumlah penderita stroke di Indonesia terus makin bertambah dan berpotensi memberikan tekanan besar terhadap sistem pelayanan kesehatan nasional. Selain itu, tingkat kematian yang diakibatkan stroke masih tergolong tinggi, menjadikannya salah satu penyebab utama kematian di Indonesia.

Masyarakat perlu memiliki pemahaman yang baik tentang berbagai faktor risikonya, serta mampu mengenali tanda-tanda peringatan dini seperti wajah yang tidak simetris, lengan yang melemah tiba-tiba, atau gangguan dalam kejelasan bicara. Namun, saat ini akses untuk mendapatkan informasi kesehatan yang akurat dan mudah dipahami masih belum merata di berbagai daerah. Hal ini seringkali berakibat pada lambatnya tindakan pencegahan atau pertolongan, yang pada akhirnya memperburuk keadaan dan hasil pemulihan pasien.

Mengingat tingkat penetrasi penggunaan ponsel pintar (*smartphone*) yang tinggi di Indonesia, teknologi *mobile health* (mHealth) menyajikan peluang yang signifikan dan sangat efisien untuk menyebarkan informasi kesehatan. Hal ini memungkinkan penyampaian informasi yang spesifik, relevan, dan edukatif, sehingga mampu menjangkau masyarakat luas hingga ke tingkat tapak bawah.

Beberapa penelitian sebelumnya telah banyak dilakukan untuk mengidentifikasi solusi digital bagi penyakit stroke, namun sebagian besar aplikasi nasional yang dikembangkan cenderung bersifat informatif satu arah atau terbatas pada manajemen pasca-stroke. Oleh karena itu, penelitian ini bergerak satu langkah lebih maju dengan menghadirkan solusi konkret melalui aplikasi "Stroke Shield". Aplikasi ini dirancang tidak hanya sebagai pusat informasi statis, tetapi sebagai alat bantu kesehatan preventif yang mengintegrasikan fitur *Screening* risiko dan senam pencegahan secara interaktif.

Dalam menghadapi tantangan kurangnya literasi dan akses pencegahan stroke, pengembangan fitur *Screening* berbasis kuesioner menjadi langkah yang relevan. Fitur deteksi dini ini memfasilitasi pengguna untuk melakukan evaluasi mandiri terhadap faktor risiko mereka secara cepat dan akurat. Sistem akan memberikan penilaian tingkat risiko secara personal, sehingga pengguna dapat lebih mawas diri dan mengambil langkah pencegahan yang tepat terhadap kondisi kesehatannya.

Selain deteksi dini, aplikasi ini menekankan tindakan preventif fisik melalui fitur senam pencegahan. Dengan memanfaatkan teknologi deteksi visual, fitur ini mampu mendeteksi dan memandu pengguna dalam melakukan gerakan-gerakan senam yang dirancang khusus untuk melatih motorik dan kelenturan tubuh sebagai upaya pencegahan stroke. Melalui penggabungan aplikasi edukasi, fitur *Screening* kesehatan, dan panduan senam pencegahan, diharapkan Stroke Shield dapat memberikan solusi yang lebih efektif, praktis, dan terjangkau dalam mendukung gaya hidup sehat dan menekan angka kejadian stroke di masyarakat.

Penggunaan *Convolutional Neural Network* (CNN) dalam aplikasi ini memiliki keunggulan dalam memproses dan mendeteksi pola visual secara *real-time*. CNN memungkinkan aplikasi untuk menganalisis tangkapan kamera dari

pergerakan pengguna dengan akurasi tinggi, mengklasifikasi postur tubuh, serta memberikan umpan balik mengenai ketepatan gerakan senam pencegahan stroke secara kontekstual. Keputusan untuk menggunakan teknologi *CNN* bertujuan untuk menciptakan pengalaman pengguna yang lebih interaktif dan memperkuat daya guna aplikasi sebagai alat panduan aktivitas fisik mandiri. Dengan demikian, penerapan *computer vision* ini diharapkan dapat memberikan solusi yang lebih efektif dan aplikatif dalam mendukung masyarakat untuk rutin melakukan tindakan pencegahan stroke.

Pengujian sistem dilakukan untuk memastikan apakah aplikasi Stroke Shield sudah bekerja dengan semestinya. Pengujian ini dilakukan dengan cara mengevaluasi tingkat akurasi antara gerakan senam aktual yang diperagakan oleh pengguna dengan hasil deteksi klasifikasi yang ditampilkan oleh sistem berbasis *CNN*, serta melakukan pengujian terhadap keakuratan prediksi pada fitur *Screening* risiko kesehatan. Secara khusus, pengujian *machine learning* pada fitur *Screening* dilakukan untuk mengevaluasi tingkat akurasi algoritma *Random Forest* dalam memproses data masukan kuesioner pengguna guna menghasilkan luaran klasifikasi tingkat risiko stroke secara presisi dan andal.

Input Kuesioner Pengguna	Hasil Sistem	Tingkat Kesesuaian
Pengguna menginput data dengan banyak faktor risiko (missal: tekanan darah tinggi, kolestrol tinggi, jarang olahraga)	Risiko Tinggi	Sesuai
Pengguna menginputkan data gaya hidup sehat tanpa riwayat penyakit atau faktor risiko	Risiko Rendah	Sesuai

Pengujian computer vision dilakukan dengan menguji keakuratan klasifikasi antara gerakan actual yang di peragakan oleh pengguna dengan hasil deteksi ditampilkan oleh sistem:

Gerakan Pengguna	Hasil Sistem	Tingkat Kesesuaian
Pengguna memperagakan instruksi gerakan senam pencegahan dengan tepat (missal: adu sisi kelingking)	Menampilkan nama gerakan secara real-time dengan status “Tepat Pose!”	Sesuai
Pengguna memperagakan gerakan yang salah atau tidak sesuai dengan dataset pelatihan	Sistem tidak mendeteksi gerakan tersebut atau menampilkan peringatan gerakan tidak sesuai	Sesuai

Pengujian ini juga dilakukan pada keseluruhan fitur yang ada untuk memastikan semua fungsi berjalan dengan baik. Pengujian aplikasi ini dilakukan menggunakan dua metode. Pertama, *Blackbox Testing*, pengujian ini berfokus pada sisi fungsionalitas dari aplikasi untuk mengevaluasi kesesuaian antara masukan (*input*) dan keluaran (*output*) pada sistem Stroke Shield. Kedua, *Usability Testing*, pengujian ini digunakan untuk mengevaluasi *User Experience* (pengalaman pengguna) dari aplikasi dengan cara melakukan pengujian langsung terhadap suatu kelompok pengguna.

Evaluasi sistem dilakukan setelah tahap pengujian selesai. Evaluasi ini dilakukan apabila terdapat ketidakakuratan atau kendala pada sistem yang berjalan saat pengujian, baik pada fungsionalitas umum, hasil prediksi *Screening*, maupun ketepatan deteksi gerakan senam.

Penelitian ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Meningkatkan kesadaran masyarakat tentang pentingnya pencegahan penyakit stroke dan pengenalan faktor risiko sejak dini melalui informasi yang interaktif dan mudah diakses.

2. Memberikan kemudahan kepada pengguna untuk mengakses beragam konten edukatif, termasuk video panduan gerakan senam pencegahan, yang bertujuan untuk membantu mereka menerapkan pola hidup sehat secara mandiri.
3. Memfasilitasi pengguna untuk melakukan deteksi dini risiko stroke melalui pengisian kuesioner *Screening* dan mendapatkan rekomendasi makanan sehat, sehingga memungkinkan pemantauan kondisi kesehatan secara berkala dan terarah.

Spesifikasi Teknis

Spesifikasi teknis meliputi:

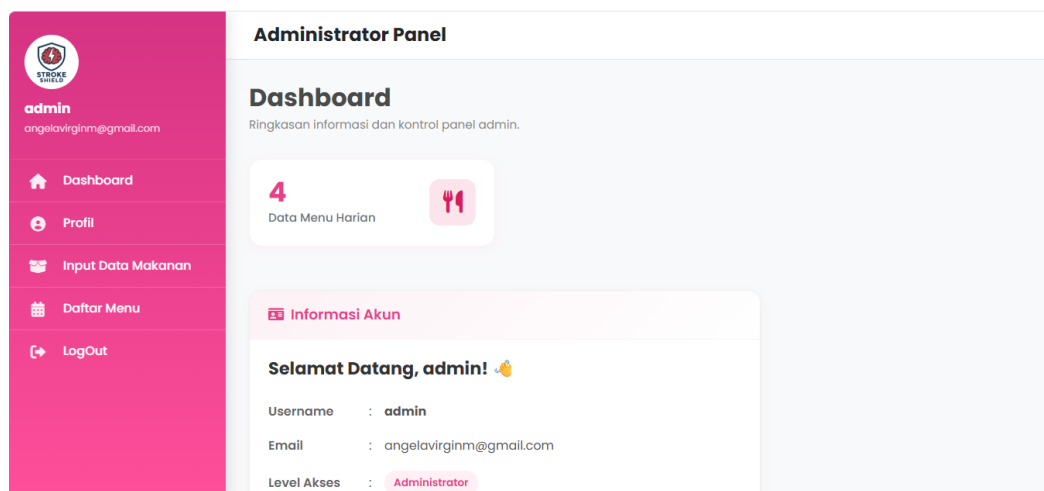
1. Modul Pengguna
2. Source Code

Berikut uraian spesifikasi untuk pembangunan aplikasi:

1. Windows 11 Ram 8gb
2. Visual Studio Code
3. Web Browser
4. Smartphone

Berikut uraian spesifikasi modul:

1. Modul untuk developer



Bagian ini hanya dapat diakses oleh developer, bertujuan untuk memudahkan melakukan perbaharuan isi konten dari aplikasi berupa rekomendasi menu harian yang ada didalamnya.

2. Modul untuk pengguna

Bagian ini adalah aplikasi Stroke Shield: Halaman ini menyajikan fitur utama yang disediakan oleh aplikasi, yaitu fitur Makanan, *Screening*, Histori, Aktivitas, Gerakan, Profile, dan Info. Selain menu utama, halaman ini juga menyediakan bagian Info Penting Stroke yang berisi sekumpulan edukatif untuk membantu pengguna memahami apa itu penyakit stroke.



Source Code

Source code back-end dari aplikasi BaikMu: Self Care mengguna *framework* Flask-API yang digunakan sebagai route untuk melakukan login, register, edit pengguna, edit password dan menambahkan serta menampilkan artikel, back-end ini juga dijadikan sebagai website untuk menampilkan landing page pada Baikmu.com serta halaman developer.

1. Import Library

```
from app import app
from flask import jsonify, request
from app.controllers.usercontroller import usercontroller
from config import Config
from pymongo import MongoClient
from app.controllers.healthcontroller import healthcontroller
```

Script tersebut adalah bagian dari program Python yang menggunakan kerangka kerja Flask dan pustaka PyMongo untuk membangun layanan *backend* API. Berikut adalah penjelasan singkat mengenai bagian-bagian impor dari *script* tersebut:

1. **from app import app:** Mengimpor instansiasi aplikasi utama (biasanya berupa objek Flask) dari modul atau *package* app lokal agar dapat digunakan untuk menjalankan server, mendaftarkan rute, atau menerapkan konfigurasi lanjutan.

2. **from flask import jsonify, request:** Mengimpor fungsi dasar dari kerangka kerja Flask. jsonify digunakan untuk memformat respons data dari server menjadi format JSON, sedangkan request digunakan untuk menangani dan mengekstrak data dari permintaan HTTP yang dikirimkan oleh klien (seperti data dari *form login* atau kuesioner *Screening*)
3. **from app.controllers.user_controller import user_controller:** Mengimpor modul pengontrol pengguna (*user_controller*) dari struktur direktori aplikasi. Bagian ini bertanggung jawab untuk mengatur logika *routing* yang berkaitan dengan manajemen pengguna, seperti proses registrasi, autentikasi (*login/logout*), dan pengelolaan profil.
4. **from config import Config:** Mengimpor kelas Config dari modul config lokal. Modul ini digunakan untuk memuat pengaturan parameter lingkungan aplikasi, seperti konfigurasi keamanan, pengaturan *port*, hingga URI koneksi *database*.
5. **from app.controllers.health_controller import health_controller:** Mengimpor modul pengontrol kesehatan (*health_controller*). Modul ini difokuskan untuk menangani logika bisnis yang berkaitan langsung dengan fitur kesehatan aplikasi, seperti memproses prediksi *Screening* risiko stroke, mengelola data rekomendasi makanan, atau menyimpan riwayat pemeriksaan.
6. **from app.controllers.health_controller import health_controller:** Mengimpor modul pengontrol kesehatan (*health_controller*). Modul ini difokuskan untuk menangani logika bisnis yang berkaitan langsung dengan fitur kesehatan aplikasi, seperti memproses prediksi *Screening* risiko stroke, mengelola data rekomendasi makanan, atau menyimpan riwayat pemeriksaan.

Setelah mengimpor semua modul yang diperlukan, script kemudian mendefinisikan dan mengkonfigurasi aplikasi Flask beserta berbagai fitur seperti CORS, SQLAlchemy untuk koneksi database, dan Mail untuk pengiriman email. Script ini kemungkinan besar akan berisi juga definisi rute-rute (*routes*) yang

menangani permintaan dari klien, termasuk operasi-operasi seperti otentikasi, manipulasi data dalam database, dan pengiriman email.

2. Konfigurasi Database

```
# MongoDB Cloud (Atlas) config - construct dari environment variable
MONGODB_USER = os.getenv("MONGODB_USER")
MONGODB_PASSWORD = os.getenv("MONGODB_PASSWORD")
MONGODB_CLUSTER = os.getenv("MONGODB_CLUSTER")
MONGODB_DATABASE = os.getenv("MONGODB_DATABASE")

# Build connection string untuk MongoDB Cloud
HOST =
f"mongodb+srv://{MONGODB_USER}:{MONGODB_PASSWORD}@
{MONGODB_CLUSTER}/{MONGODB_DATABASE}?retryWrites=true&w=majority"
DATABASE = MONGODB_DATABASE

# Inisialisasi MongoDB client dan koleksi
client = MongoClient(HOST)
db = client[DATABASE]
menu_collection = db["input_data"] # gunakan koleksi 'input_data'

# Konfigurasi Email
MAIL_SERVER = 'smtp.gmail.com'
MAIL_PORT = 587
MAIL_USE_TLS = True
MAIL_USERNAME = os.getenv('EMAIL_USER') # ambil dari .env
MAIL_PASSWORD = os.getenv('EMAIL_PASS') # ambil dari .env
MAIL_DEFAULT_SENDER = os.getenv('EMAIL_USER')

# Google OAuth
GOOGLE_CLIENT_ID ="636011420760-
j38mr8soc86ulukhjgicfhce7co4q7d.apps.googleusercontent.com"
```

Script di atas melakukan konfigurasi untuk koneksi database dan layanan pihak ketiga yang akan digunakan oleh aplikasi. Konfigurasi yang dilakukan termasuk:

1. Konfigurasi database menggunakan MongoDB Cloud (Atlas):
 - MONGODB_USER, MONGODB_PASSWORD, MONGODB_CLUSTER, MONGODB_DATABASE: Mengekstrak kredensial dan informasi *database* dari variabel lingkungan (*environment*)

variables) menggunakan `os.getenv()` guna menjaga keamanan data sensitif.

- **HOST:** Membangun *connection string* (URI) dengan format `mongodb+srv://` yang secara aman mengarah ke *cluster* MongoDB Cloud.
- **client, db, menu_collection:** Menginisialisasi koneksi `MongoClient`, memilih *database*, dan menentukan koleksi `input_data` yang akan digunakan aplikasi untuk menyimpan atau memproses data.

2. Konfigurasi pengiriman email:

- **MAIL_SERVER, MAIL_PORT, MAIL_USE_TLS:** Mendefinisikan pengaturan server SMTP Gmail ('smtp.gmail.com') pada port 587, serta mengaktifkan fitur enkripsi TLS untuk memastikan koneksi pengiriman email berjalan dengan aman.
- **MAIL_USERNAME, MAIL_PASSWORD:** Mendefinisikan kredensial pengguna Gmail yang diambil dari berkas `.env` untuk melakukan otentikasi saat mengirim email (misalnya, untuk tautan verifikasi atau OTP).
- **MAIL_DEFAULT_SENDER:** Mendefinisikan alamat pengirim *default* yang akan muncul saat email diterima oleh pengguna.
- Konfigurasi Google OAuth:
 - **GOOGLE_CLIENT_ID:** Mendefinisikan ID Klien Google yang digunakan untuk memvalidasi dan memproses autentikasi apabila aplikasi menyediakan fitur pendaftaran atau masuk (*login*) menggunakan akun Google.

Dengan konfigurasi ini, *system backend* telah siap digunakan dengan koneksi database NoSQL MongoDB yang berjalan di cloud, layanan pengiriman email via SMTP Gmail, serta dukungan kunci kredensial untuk fitur autentikasi Google.

3. Route Register

```
@app.route("/api/register", methods=["POST"])
def register():
    return usercontroller.register()
```

```

def register():
    data = request.get_json()
    username = data.get("nama")
    email = data.get("email")
    password = data.get("password")
    role = data.get("role", "user")

    if not username or not email or not password:
        return jsonify({"status": "error", "message": "Field tidak lengkap"}), 400

    if db.user.find_one({"email": email}):
        return jsonify({"status": "error", "message": "Email sudah terdaftar"}), 400

    hashed_pw = generate_password_hash(password)
    new_user = {
        "nama": username,
        "email": email,
        "password": hashed_pw,
        "role": role,
        "confirmed": False
    }

    db.user.insert_one(new_user)
    usercontroller.send_otp(email, username)

    return jsonify({
        "status": "success",
        "message": "Registrasi berhasil. OTP telah dikirim ke email untuk verifikasi."
    }), 201

```

Dalam bagian kode tersebut, terdapat sebuah *endpoint* API `/api/register` yang digunakan untuk mendaftarkan pengguna baru ke dalam sistem. Ketika sebuah permintaan POST diterima ke *endpoint* ini, data dalam format JSON yang diperlukan seperti nama, email, password, serta role (yang secara *default* diatur sebagai "user") diambil dari permintaan tersebut. Selanjutnya, dilakukan pengecekan apakah kolom data utama (nama, email, dan password) sudah diisi secara lengkap oleh pengguna. Jika tidak lengkap, pesan kesalahan "Field tidak lengkap" akan dikirimkan kembali ke pengguna dengan status kode 400..

Setelah itu, dilakukan pengecekan ke dalam database MongoDB untuk melihat apakah email yang dimasukkan oleh pengguna sudah terdaftar sebelumnya di dalam koleksi `user`. Jika email sudah terdaftar, pesan kesalahan "Email sudah terdaftar" akan dikirimkan kembali ke pengguna, juga dengan status kode 400.

Jika semua pengecekan validasi telah berhasil dilewati, data pengguna baru akan disiapkan untuk dimasukkan ke dalam database. Password yang dimasukkan akan di-hash terlebih dahulu menggunakan fungsi `generate_password_hash` untuk memastikan keamanan kredensial pengguna. Selanjutnya, pengguna akan ditandai sebagai belum terverifikasi (`"confirmed": False`), karena mereka masih harus melakukan tahap verifikasi email. Data pengguna dalam bentuk dokumen (dictionary) ini kemudian disimpan ke dalam database MongoDB menggunakan perintah `insert_one()`. Setelah data tersimpan, sistem akan memanggil fungsi `send_otp` untuk mengirimkan kode OTP ke email pengguna yang didaftarkan.

Jika semua langkah tersebut berhasil dilakukan, respons sukses akan dikirimkan kembali ke pengguna dengan pesan bahwa registrasi telah berhasil dan OTP telah dikirim ke email untuk tahap verifikasi. Respons ini mengembalikan status kode 201 yang secara spesifik menunjukkan bahwa permintaan telah berhasil diproses dan sebuah *resource* atau data baru telah berhasil dibuat di dalam server.

4. Route Login

```
@app.route("/api/login", methods=["POST"])
def login():
    return usercontroller.login()
def login():
    data = request.get_json()
    email = data.get("email")
    password = data.get("password")

    user = db.user.find_one({"email": email})
    if not user or not check_password_hash(user["password"], password):
        return jsonify({'status': "error", "message": "Email atau password salah"}),
401

    access_token = generate_jwt(str(user["_id"]), user["role"])

    user_data = {
        "id": str(user["_id"]),
        "nama": user["nama"],
```

```

        "email": user["email"],
        "access_token": access_token,
        "foto_profil": user.get("foto_profil", "")
    }

    return jsonify({"status": "success", "data": user_data}), 200

```

Pada endpoint `/api/login`, aplikasi mengizinkan pengguna untuk masuk ke dalam system. Saat menerima permintaan POST, data yang dibutuhkan seperti alamat email dan kata sandi diambil dari permintaan dalam format JSON.

yang terdaftar dengan alamat email yang diberikan, pesan kesalahan akan dikirimkan kembali ke pengguna dengan status kode 400.

Setelah itu, dilakukan pencarian pengguna berdasarkan alamat email yang diberikan di dalam *database*. Sistem kemudian akan langsung memvalidasi apakah pengguna dengan email tersebut ditemukan dan apakah kata sandi yang dimasukkan sesuai dengan yang disimpan dalam *database* menggunakan bantuan fungsi `check_password_hash`. Jika tidak ada pengguna yang terdaftar dengan alamat email yang diberikan, atau jika kata sandi yang dimasukkan tidak cocok, pesan kesalahan "Email atau password salah" akan dikirimkan kembali ke pengguna dengan status kode 401.

Jika alamat email ditemukan dan kata sandi yang dimasukkan sesuai, maka sistem akan *men-generate* token akses berupa JWT (*JSON Web Token*) menggunakan fungsi `generate_jwt` yang berisi informasi ID pengguna serta peran (*role*) mereka. Selanjutnya, sistem akan menyusun sekumpulan data pengguna (*user_data*) yang mencakup ID, nama, alamat email, token akses yang baru dibuat, serta foto profil pengguna. Respons sukses kemudian dikirimkan kembali ke pengguna dengan membawa data tersebut. Respons ini memiliki status kode 200 yang menunjukkan bahwa proses *login* telah berhasil diproses tanpa ada kesalahan.

5. Route Verify Otp

```

@app.route("/api/verify-otp", methods=["POST"])
def verify_otp():
    return usercontroller.verify_otp()
def verify_otp():
    data = request.get_json()

```

```

email = data.get("email")
otp_input = data.get("otp")

if not email or not otp_input:
    return jsonify({"status": "error", "message": "Email dan OTP wajib diisi"}),
400

otp_data = db.otp.find_one({"email": email}, sort=[("expires_at", -1)])
if not otp_data:
    return jsonify({"status": "error", "message": "Kode OTP tidak ditemukan"}),
404

now = datetime.utcnow()
if now > otp_data["expires_at"]:
    return jsonify({"status": "error", "message": "Kode OTP sudah kadaluarsa"}),
400

if otp_input != otp_data["otp"]:
    return jsonify({"status": "error", "message": "Kode OTP salah"}), 400

db.user.update_one({"email": email}, {"$set": {"confirmed": True}})
return jsonify({"status": "success", "message": "Email berhasil diverifikasi"}),
200

```

Pada endpoint `/api/verify-otp`, aplikasi menyediakan fitur untuk memverifikasi akun pengguna yang baru mendaftar menggunakan kode OTP (*One-Time Password*). Saat menerima permintaan POST, data yang dibutuhkan seperti alamat email dan kode OTP yang diinputkan diambil dari permintaan tersebut dalam format JSON. Sistem kemudian melakukan pengecekan awal untuk memastikan bahwa kedua data tersebut telah diisi. Jika alamat email atau kode OTP kosong, pesan kesalahan "Email dan OTP wajib diisi" akan dikirimkan kembali ke pengguna dengan status kode 400.

Jika data ditemukan, sistem akan melanjutkan dengan dua tahap validasi lanjutan. Pertama, sistem mengecek apakah waktu saat ini sudah melewati batas waktu kedaluarsa kode OTP tersebut. Jika sudah lewat, pesan "Kode OTP sudah kadaluarsa" akan dikirimkan dengan status kode 400. Kedua, sistem mencocokkan kode OTP yang dimasukkan oleh pengguna dengan kode yang tersimpan di dalam

database. Jika kodenya tidak sama, pesan "Kode OTP salah" akan dikembalikan ke pengguna dengan status kode 400.

Jika semua tahapan validasi tersebut berhasil dilewati dan kode OTP dinyatakan valid, sistem akan memperbarui data pengguna di dalam koleksi *user*. Status pengguna akan diubah menjadi terverifikasi dengan memperbarui nilai "confirmed" menjadi *True* menggunakan perintah *update_one()*. Sebagai langkah terakhir, respons sukses akan dikirimkan kembali ke pengguna dengan pesan "Email berhasil diverifikasi". Respons ini memiliki status kode 200 yang menunjukkan bahwa proses verifikasi akun telah berhasil diselesaikan tanpa ada kesalahan.

6. Route Edit Password

```
@app.route("/api/forgot-password", methods=["POST"])
def forgot_password():
    return usercontroller.send_reset_otp()
def reset_password():
    data = request.get_json()
    email = data.get("email")
    new_password = data.get("new_password")

    if not email or not new_password:
        return jsonify({"status": "error", "message": "Email dan password wajib
diisi"}), 400

    hashed_pw = generate_password_hash(new_password)
    result = db.user.update_one({"email": email}, {"$set": {"password":
hashed_pw}})

    if result.modified_count == 0:
        return jsonify({"status": "error", "message": "Gagal memperbarui
password"}), 400

    return jsonify({"status": "success", "message": "Password berhasil direset"}),
200
```

Pada endpoint */forgot-password*, aplikasi menyediakan fitur pemulihan kata sandi (yang diwakili oleh fungsi *reset_password*) untuk memungkinkan pengguna memperbarui kata sandi mereka. Saat menerima permintaan *POST*, data yang

dibutuhkan seperti alamat email dan kata sandi baru (`new_password`) diambil dari permintaan tersebut dalam format JSON.

Sistem kemudian mengecek hasil dari operasi pembaruan tersebut melalui nilai `result.modified_count`. Jika nilainya adalah 0 (yang berarti tidak ada data yang berubah, bisa jadi karena email tidak ditemukan di dalam *database*), maka pesan kesalahan "Gagal memperbarui password" akan dikirimkan kepada pengguna dengan status kode 400.

Setelah tahap validasi berhasil dilewati, kata sandi baru yang dimasukkan akan dienkripsi atau di-hash terlebih dahulu menggunakan fungsi `generate_password_hash` untuk memastikan keamanan kredensial pengguna. Selanjutnya, sistem akan melakukan pencarian pengguna berdasarkan alamat email di dalam *database* (koleksi `user`) dan langsung memperbarui field kata sandinya dengan *hash* yang baru menggunakan perintah `update_one()`.

Sistem kemudian mengecek hasil dari operasi pembaruan tersebut melalui nilai `result.modified_count`. Jika nilainya adalah 0 (yang berarti tidak ada data yang berubah, bisa jadi karena email tidak ditemukan di dalam *database*), maka pesan kesalahan "Gagal memperbarui password" akan dikirimkan kepada pengguna dengan status kode 400.

Jika operasi pembaruan berhasil dan ada data yang diubah, respons sukses akan dikirimkan kembali ke pengguna dengan pesan "Password berhasil direset". Respons ini mengembalikan status kode 200 yang menunjukkan bahwa proses pembaruan kata sandi telah berhasil diselesaikan tanpa ada kesalahan di sisi *server*.

7. Route Login Google

```
@app.route("/api/auth/google/flutter", methods=["POST"])
def auth_google_flutter():
    print("Route /api/auth/google/flutter dipanggil")
    return usercontroller.login_google_flutter()
def login_google_flutter():
    data = request.get_json()
    token_google = data.get("idToken")

    if not token_google:
        return jsonify({"status": "error", "message": "ID
Token tidak ditemukan"}), 400

    try:
        id_info =
id_token.verify_oauth2_token(token_google, requests.Request(),
GOOGLE_CLIENT_ID)

        email = id_info.get("email")
        nama = id_info.get("name", "User Google")
        google_id = id_info.get("sub")
        foto_profil = id_info.get("picture")

        if not email:
            return jsonify({"status": "error", "message":
"Email tidak ditemukan dalam token"}), 400

        user = db.user.find_one({"email": email})
        if not user:
            user = {
                "nama": nama,
                "email": email,
                "password": None,
                "role": "user",
                "confirmed": True,
                "google_id": google_id,
                "foto_profil": foto_profil
            }
            db.user.insert_one(user)
        else:
            db.user.update_one({"email": email}, {
                "$set": {
                    "confirmed": True,
                    "google_id": google_id,
                    "foto_profil": foto_profil
```

```

        })

        access_token = generate_jwt(str(user["_id"]),
user["role"])

        return jsonify({
            "status": "success",
            "data": {
                "id": str(user["_id"]),
                "nama": user["nama"],
                "email": user["email"],
                "foto_profil": user.get("foto_profil",
""),
                "access_token": access_token
            }
        }), 200

    except ValueError as e:
        return jsonify({"status": "error", "message":
f"Token tidak valid: {str(e)}"}), 401

```

Pada *endpoint* `/api/auth/google/flutter`, aplikasi menyediakan fitur autentikasi (*Single Sign-On*) yang memungkinkan pengguna untuk mendaftar atau masuk secara praktis menggunakan akun Google mereka melalui aplikasi *mobile* berbasis Flutter.

Saat menerima permintaan POST, sistem pertama-tama mengambil data `idToken` (token identitas dari Google) yang dikirimkan oleh klien dalam format JSON. Jika token tersebut kosong atau tidak disertakan, pesan kesalahan "ID Token tidak ditemukan" akan langsung dikirimkan kembali ke pengguna dengan status kode 400.

Selanjutnya, di dalam blok `try-except`, sistem memverifikasi keabsahan dan integritas token tersebut menggunakan fungsi `verify_oauth2_token` dari pustaka Google Auth. Proses ini mencocokkan token dengan `GOOGLE_CLIENT_ID` milik aplikasi. Jika token valid, sistem akan mengekstrak informasi profil dari akun Google pengguna, seperti alamat email, nama, ID unik Google (`sub`), dan URL foto profil. Namun, jika token terdeteksi tidak valid (misalnya karena sudah

kedaluwarsa atau dimanipulasi), sistem akan menangkap pengecualian (*exception*) berupa `ValueError` dan mengembalikan pesan "Token tidak valid" dengan status kode 401.

Setelah informasi berhasil diekstrak dan alamat email dipastikan tersedia, sistem akan mencari apakah pengguna dengan email tersebut sudah terdaftar di dalam *database* (koleksi `user`):

- **Jika belum terdaftar:** Sistem akan secara otomatis membuat akun baru untuk pengguna tersebut. Data yang disimpan mencakup nama dan email dari Google, kata sandi yang dikosongkan (`password: None`), serta status akun yang langsung ditandai sebagai terverifikasi (`"confirmed": True`). Data ini kemudian dimasukkan ke *database* menggunakan perintah `insert_one()`.
- **Jika sudah terdaftar:** Sistem cukup memperbarui profil pengguna yang ada dengan menyinkronkan status verifikasi, ID Google, dan foto profil terbaru menggunakan perintah `update_one()`.

Sebagai langkah terakhir, sistem men-*generate* token akses (JWT) baru berdasarkan ID dan peran (*role*) pengguna. Respons sukses kemudian dikirimkan kembali ke aplikasi klien dengan status kode 200, membawa data pengguna yang mencakup ID, nama, email, foto profil, beserta token akses yang akan digunakan untuk mengamankan sesi aplikasi selanjutnya.

8. Route *Screening History*

```
@healthcontroller.route('/Screening-history/<user_id>',
methods=['GET'])
def get_Screening_history(user_id):
    try:
        data = list(db.Screening.find({"user_id":
user_id}).sort("tanggal", -1))

        for item in data:
            item["_id"] = str(item["_id"])
            item["tanggal"] = item["tanggal"].isoformat()

        return jsonify({
            "status": "success",
            "data": data
```

```

    }), 200
except Exception as e:
    return jsonify({"status": "error", "message": f"Gagal
mengambil data: {str(e)}"}), 500

```

Pada *endpoint* `/Screening-history/<user_id>`, aplikasi menyediakan fitur untuk mengambil riwayat hasil *Screening* risiko stroke milik pengguna tertentu. Saat menerima permintaan GET dari klien, sistem akan menangkap parameter `user_id` secara langsung dari URL untuk mengidentifikasi data riwayat milik siapa yang perlu ditarik.

Di dalam blok `try-except` untuk menjaga stabilitas sistem dari kemungkinan *error*, sistem melakukan pencarian seluruh data riwayat pada *database* MongoDB (di dalam koleksi *Screening*) yang memiliki kecocokan dengan `user_id` tersebut. Data yang ditemukan kemudian diurutkan berdasarkan atribut `"tanggal"` secara menurun (`-1`), sehingga pengguna akan menerima hasil *Screening* yang paling baru di urutan teratas.

Setelah data didapatkan dan diubah ke dalam bentuk daftar (*list*), sistem melakukan iterasi pada setiap item data untuk memformat ulang beberapa atribut agar dapat dikonversi ke dalam bentuk JSON dengan baik. Format ulang ini meliputi pengubahan ID unik bawaan MongoDB (`"_id"`) menjadi teks (*string*) dan pengubahan format objek waktu (`tanggal`) menjadi format standar internasional ISO 8601 menggunakan fungsi `isoformat()`.

Jika seluruh proses pengambilan dan pemformatan data ini berhasil, respons sukses akan dikirimkan kembali ke klien dengan membawa susunan data riwayat tersebut. Respons ini mengembalikan status kode 200 yang menunjukkan bahwa permintaan GET telah berhasil diproses tanpa kendala. Namun, jika terjadi kesalahan atau anomali pada *server* selama proses berlangsung (misalnya gangguan koneksi *database*), sistem akan menangkap pengecualian (*exception*) dan mengembalikan pesan "Gagal mengambil data" beserta detail kesalahannya, diiringi dengan status kode 500 yang menandakan terjadinya *Internal Server Error*.

9. Route *Screening*

```

@healthcontroller.route('/deteksi-stroke', methods=['POST'])
def deteksi_stroke():
    if model is None or encoder is None:

```

```

        return jsonify({
            "status": "error",
            "message": "Model atau Encoder tidak ditemukan.
Hubungi administrator."
        }), 500

    data = request.get_json()

    # Ambil user_id dari request
    user_id = data.get("user_id")
    if not user_id:
        return jsonify({"status": "error", "message": "user_id
wajib disertakan"}), 400

    # Validasi field wajib (sesuai training model)
    required_fields = [
        'jenis_kelamin', 'umur', 'tinggi_badan',
'berat_badan',
        'tekanan_darah', 'kolesterol', 'gula_darah',
        'riwayat_keluarga', 'merokok', 'aktivitas_fisik',
'pola_makan'
    ]

    for field in required_fields:
        if field not in data:
            return jsonify({"status": "error", "message":
f"Field '{field}' wajib diisi"}), 400

    try:
        # Siapkan data input sesuai dengan format training
        data_baru = {
            'jenis_kelamin': data.get('jenis_kelamin'),
            'umur': int(data.get('umur')),
            'tinggi_badan': float(data.get('tinggi_badan')),
            'berat_badan': float(data.get('berat_badan')),
            'tekanan_darah': float(data.get('tekanan_darah')),
            'kolesterol': float(data.get('kolesterol')),
            'gula_darah': float(data.get('gula_darah')),
            'riwayat_keluarga': data.get('riwayat_keluarga'),
            'merokok': data.get('merokok'),
            'aktivitas_fisik': data.get('aktivitas_fisik'),
            'pola_makan': data.get('pola_makan')
        }

        # Ubah ke DataFrame

```

```

df_input = pd.DataFrame([data_baru])

# Encoding sesuai yang disimpan (untuk categorical
columns)
for col in encoder:
    if col in df_input.columns:
        df_input[col] =
df_input[col].map(encoder[col])

# Lakukan prediksi
prediction = model.predict(df_input)

# Tentukan hasil (sesuai training: 1 = TINGGI, 0 =
RENDAH)
hasil = "● Risiko Stroke: TINGGI" if prediction[0] ==
1 else "□ Risiko Stroke: RENDAH"
confidence =
float(model.predict_proba(df_input)[0].max() * 100) if
hasattr(model, 'predict_proba') else None

# Simpan data input + hasil ke MongoDB
Screening_data = {
    "user_id": user_id,
    "input": {
        "jenis_kelamin": data.get('jenis_kelamin'),
        "umur": int(data.get('umur')),
        "tinggi_badan":
float(data.get('tinggi_badan')),
        "berat_badan": float(data.get('berat_badan')),
        "tekanan_darah":
float(data.get('tekanan_darah')),
        "kolesterol": float(data.get('kolesterol')),
        "gula_darah": float(data.get('gula_darah')),
        "riwayat_keluarga":
data.get('riwayat_keluarga'),
        "merokok": data.get('merokok'),
        "aktivitas_fisik":
data.get('aktivitas_fisik'),
        "pola_makan": data.get('pola_makan')
    },
    "hasil": hasil,
    "confidence": confidence,
    "tanggal": datetime.utcnow()
}

```

```

db.Screening.insert_one(Screening_data)

response = {
    "status": "success",
    "hasil": hasil,
    "confidence": confidence
}

return jsonify(response), 200

except Exception as e:
    return jsonify({
        "status": "error",
        "message": f"Error dalam pemrosesan data: {str(e)}"
    }), 400

```

Pada *endpoint* /deteksi-stroke, aplikasi menyediakan fungsionalitas inti untuk melakukan *Screening* atau deteksi dini risiko penyakit stroke berbasis *machine learning*. Saat menerima permintaan POST, sistem pertama-tama akan memastikan bahwa model kecerdasan buatan dan *encoder* (pemroses data kategori) telah termuat dengan baik di dalam *server*. Jika komponen tersebut tidak ditemukan, pesan kesalahan akan dikembalikan dengan status kode 500.

Sistem kemudian mengambil data masukan dari kuesioner pengguna dalam format JSON. Proses validasi ketat dilakukan untuk memastikan ID pengguna (*user_id*) dan seluruh parameter kesehatan yang wajib (seperti umur, tinggi badan, berat badan, tekanan darah, kolesterol, gula darah, riwayat keluarga, status merokok, aktivitas fisik, dan pola makan) telah diisi secara lengkap. Jika ada satu saja metrik kesehatan yang terlewat, pesan kesalahan spesifik akan dikirimkan dengan status kode 400.

Di dalam blok *try-except* untuk penanganan proses komputasi, data yang valid tersebut diproses lebih lanjut. Sistem akan mengonversi tipe data dari klien agar seragam (misalnya memastikan nilai kolesterol berupa angka desimal atau *float*) dan menyusunnya ke dalam bentuk struktur data Pandas *DataFrame*. Untuk data yang bersifat kategori atau pilihan ganda, sistem menggunakan *encoder* untuk

memetakan teks tersebut menjadi format angka yang dapat diproses oleh algoritma *machine learning*.

Setelah data diformat sesuai dengan standar pelatihan, model *machine learning* (seperti algoritma *Random Forest*) akan melakukan prediksi klasifikasi. Hasil prediksi tersebut kemudian diterjemahkan ke dalam bentuk teks yang mudah dipahami, yaitu "● Risiko Stroke: TINGGI" (jika model menghasilkan angka 1) atau "□ Risiko Stroke: RENDAH" (jika model menghasilkan angka 0). Sistem juga berupaya menghitung tingkat persentase probabilitas (*confidence score*) dari prediksi tersebut.

Untuk keperluan rekam medis digital pengguna, sistem kemudian mengemas seluruh data yang terdiri dari ID pengguna, riwayat input kuesioner awal, hasil prediksi, tingkat probabilitas, beserta stempel waktu (*timestamp*) pencatatan, lalu menyimpannya secara permanen ke dalam *database* MongoDB pada koleksi *Screening* menggunakan perintah `insert_one()`.

Sebagai langkah terakhir, respons sukses yang membawa hasil prediksi dan nilai probabilitas tersebut akan dikirimkan kembali ke aplikasi pengguna dengan status kode 200. Apabila terjadi kegagalan sistematis atau kesalahan konversi tipe data selama proses analisis berjalan, sistem akan langsung menangkap *error* tersebut dan mengembalikan pesan "Error dalam pemrosesan data" kepada pengguna dengan status kode 400.

10. Route Senam

```
@healthcontroller.route('/senam-history', methods=['POST'])
def add_senam_history():
    data = request.get_json()
    user_id = data.get("user_id")
    gerakan_name = data.get("gerakan_name")
    akurasi = data.get("akurasi")
    waktu = data.get("waktu")

    if not user_id or not gerakan_name or akurasi is None or waktu is None:
        return jsonify({"status": "error", "message": "Data user_id, gerakan_name, akurasi, dan waktu wajib diisi"}), 400

    try:
```

```

        history_data = {
            "user_id": user_id,
            "gerakan_name": gerakan_name,
            "akurasi": float(akurasi),
            "waktu": int(waktu),
            "tanggal": datetime.utcnow()
        }
        db.senam_history.insert_one(history_data)
        return jsonify({"status": "success", "message":
            "Histori senam berhasil disimpan"}), 200
    except Exception as e:
        return jsonify({"status": "error", "message": f"Gagal
            menyimpan data: {str(e)}"}), 500

```

Pada *endpoint* /senam-history, aplikasi menyediakan fitur untuk mencatat dan menyimpan riwayat aktivitas senam pencegahan stroke yang telah dilakukan oleh pengguna. Saat menerima permintaan POST, sistem pertama-tama akan mengekstrak data dari aplikasi klien dalam format JSON. Data yang dibutuhkan mencakup ID pengguna (*user_id*), nama gerakan senam yang dieksekusi (*gerakan_name*), tingkat akurasi gerakan yang dideteksi oleh model *CNN* (*akurasi*), serta durasi waktu pelaksanaannya (*waktu*).

Sistem kemudian melakukan validasi awal secara ketat untuk memastikan bahwa keempat parameter utama tersebut telah dikirimkan oleh klien. Jika ada salah satu data yang kosong atau tidak disertakan (terutama nilai akurasi dan waktu yang rentan bernilai *None*), sistem akan langsung menolak permintaan tersebut dan mengembalikan pesan kesalahan "Data user_id, gerakan_name, akurasi, dan waktu wajib diisi" dengan status kode 400.

Di dalam blok *try-except* untuk menjaga keandalan sistem, data yang sudah divalidasi tersebut diproses lebih lanjut. Sistem memastikan tipe data sudah sesuai dengan mengonversi nilai akurasi menjadi angka desimal (*float*) dan durasi waktu menjadi bilangan bulat (*integer*). Data ini kemudian digabungkan dengan stempel waktu server saat ini (*datetime.utcnow()*) sebagai penanda kapan senam tersebut dilakukan. Seluruh informasi ini dikemas ke dalam satu struktur dokumen lalu disimpan secara permanen ke dalam *database* MongoDB pada koleksi *senam_history* menggunakan perintah *insert_one()*.

Sebagai langkah terakhir, jika proses penyimpanan ke *database* berhasil dilakukan, respons sukses akan dikirimkan kembali ke aplikasi pengguna dengan pesan "Histori senam berhasil disimpan" beserta status kode 200. Namun, apabila terjadi anomali atau kegagalan teknis saat proses penyimpanan (seperti terputusnya koneksi *database*), sistem akan menangkap *error* tersebut dan mengembalikan pesan "Gagal menyimpan data" beserta detail teknisnya, diiringi dengan status kode 500 yang menandakan terjadinya *Internal Server Error*.