LAMPIRAN

Lampiran 1 Surat Kesediaan Bimbingan Skripsi

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	:	Hasnita Rani Kumala
NIM	:	21090092
Program Studi	:	Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	:	Dwi Intan Af'idah, S.T., M.Kom.
Status	:	Dosen
NIDN	:	0620089203
Jabatan Fungsional	:	Lektor
Pangkat/Golongan	:	Penata, III/c

Pada hari ini Rabu tanggal 06 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I Skripsi Pihak Pertama dengan syarat adanya peningkatan progres dari proyek skripsi yang dilaporkan setiap satu pekan. Jika Pihak 1 tidak memenuhi syarat tersebut, maka Pihak 2 berhak memutuskan kesepakatan ini. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 06 Maret 2025

Pihak Pertama	Pihak Kedua
Hasnita Rani Kumala	Dwi Intan Af'idah, S.T., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

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

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	: Hasnita Rani Kumala
NIM	: 21090092
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Hepatika Zidny Ilmadina, S.Pd., M.Kom.
Status	: Dosen Tetap
NIDN	: 0618119101
Jabatan Fungsional	: Asisten Ahli
Pangkat/Golongan	: Penata muda tk. I/ III-b

Pada hari ini Rabu tanggal 06 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing II Skripsi Pihak Pertama dengan syarat Pihak Pertama wajib melakukan bimbingan Skripsi minimal 8 kali kepada Pihak Kedua. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 06 Maret 2025

Pihak Pertama	Pihak Kedua
Hasnita Rani Kumala	Hepatika Zidny Ilmadina, S.Pd., M.Kom.

Mengetahui
Ketua Program Studi Sarjana Terapan Teknik Informatika

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

Lampiran 2 Surat Pernyataan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. Nama : Hasnita Rani Kumala
   Kewarganegaraan : Indonesia
   Alamat : Jl. Sunan Gunung Jati 1. RT/RW 001/002, Kelurahan Limbangan Wetan, Kecamatan Brebes, Provinsi Jawa Tengah
2. Nama : Dwi Intan Afidah, S.T., M.Kom
   Kewarganegaraan : Indonesia
   Alamat : Desa Grinting RT/RW 003/001, Kecamatan Bulakamba, Kabupaten Brebes, Provinsi Jawa Tengah
3. Nama : Hepatika Zidny Ilmadina, S.Pd., M.Kom
   Kewarganegaraan : Indonesia
   Alamat : Jalan Kenanga Gang I Nomor 9, Kelurahan Mangkukusuman, Kecamatan Tegal Timur, Provinsi Jawa Tengah

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : Aplikasi Panduan Gerakan Terapi untuk Mengurangi Nyeri Punggung Bawah Berbasis Mobile
   - 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 berperkara 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, 13 Juni 2025

(Hasnita Rani Kumala)
Pemegang Hak Cipta *

(Dwi Intan Afidah, S.T., M.Kom)
Pemegang Hak Cipta *

(Hepatika Zidny Ilmadina, S.Pd., M.Kom)
Pemegang Hak Cipta *

* Semua pemegang hak cipta agar menandatangani di atas materai.

Lampiran 3 Surat Pengalihan HKI

SURAT PENGALIHAN HAK CIPTA

Yang bertanda tangan di bawah ini :

1.	N a m a	: Hasnita Rani Kumala
	Kewarganegaraan	: Indonesia
	Alamat	: Jl. Sunan Gunung Jati 1. RT/RW 001/002, Kelurahan Limbangan Wetan, Kecamatan Brebes, Provinsi Jawa Tengah
2.	N a m a	: Dwi Intan Afidah, S.T., M.Kom
	Kewarganegaraan	: Indonesia
	Alamat	: Desa Grinting RT/RW 003/001, Kecamatan Bulakamba, Kabupaten Brebes, Provinsi Jawa Tengah
3.	N a m a	: Hepatika Zidny Ilmadina, S.Pd., M.Kom
	Kewarganegaraan	: Indonesia
	Alamat	: Jalan Kenanga Gang I Nomor 9, Kelurahan Mangkukusuman, Kecamatan Tegal Timur

Adalah **Pihak I** selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :

N a m a	: Pusat Penelitian dan Pengabdian Masyarakat (P3M) Politeknik Harapan Bersama
Alamat	: Jl. Mataram No.9 Pesurungan Lor Kota Tegal

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa Program Komputer dengan judul "*Aplikasi Panduan Gerakan Terapi untuk Mengurangi Nyeri Punggung Bawah Berbasis Mobile*" untuk didaftarkan di Direktorat Hak Cipta dan Desain Industri, Direktorat Jenderal Kekayaan Intelektual, Kementerian Hukum dan Hak Asasi Manusia Republik Indonesia.

Demikianlah surat pengalihan hak ini kami buat, agar dapat dipergunakan sebagaimana mestinya.

Tegal,13 Juni 2025

Pemegang Hak Cipta
Kepala P3M

(Muhammad Fikri Hidayattullah, S.T., M.Kom.)

Pencipta

(Hasnita Rani Kumala)

(Dwi Intan Afidah, S.T., M.Kom)

(Hepatika Zidny Ilmadina, S.Pd., M.Kom)

Lampiran 4 Surat Observasi Fisioterapi

POLITEKNIK HARAPAN BERSAMA
The True Vocational Campus

Sarjana Terapan Teknik Informatika

Nomor : 15.03/TI.PHB/III/2025
Lampiran : -
Hal : Permohonan Izin Observasi
Kepada Yth. : **Kepala RSUD Brebes**
Di Tempat

Dengan hormat, mahasiswa dengan identitas berikut ini:

nama : Hasnita Rani Kumala
NIM : 21090092
prodi : Sarjana Terapan Teknik Informatika

Bermaksud melakukan penelitian untuk keperluan Skripsi dengan judul "**Aplikasi Panduan Gerakan Terapi untuk Mencegah dan Mengurangi Nyeri Punggung Bawah Berbasis Mobile**". Kami memohon Bapak/Ibu memberikan izin kepada mahasiswa yang bersangkutan agar memperoleh data, keterangan, dan bahan yang diperlukan.

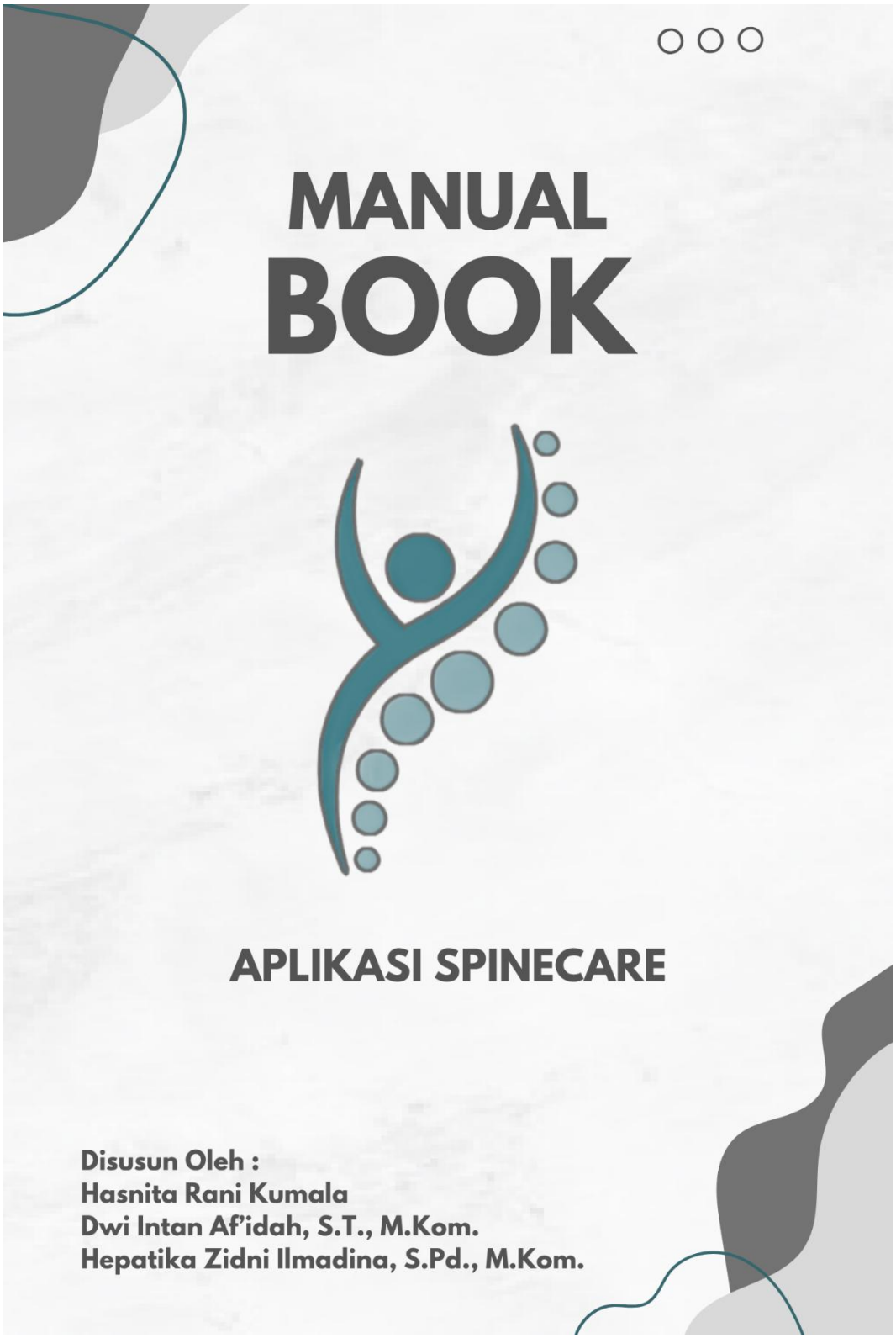
Demikian permohonan ini disampaikan, atas perhatian kami ucapkan terima kasih.

Tegal, 17 Maret 2025
Ka. Prodi S.Tr. Teknik Inormatika,

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

Jl. Mataram No. 9 Kota Tegal 52143, Jawa Tengah, Indonesia.
(0283)352000
informatika@poltektegal.ac.id
poltektegal.ac.id

Lampiran 5 Manual Book Aplikasi



PENDAHULUAN

1. Tujuan Pembuatan Dokumen

Dokumen ini disusun sebagai panduan lengkap bagi pengguna aplikasi SpineCare. Tujuannya adalah untuk membantu pengguna memahami seluruh fitur yang tersedia dan cara mengoperasikannya dengan tepat, mulai dari proses instalasi, pendaftaran akun, hingga pelaksanaan sesi terapi mandiri.

2. Nilai Inovasi

SpineCare menawarkan pendekatan baru dalam terapi mandiri berbasis teknologi. Tidak hanya menyajikan video latihan seperti aplikasi sejenis, Spinecare dilengkapi fitur asesmen awal untuk menyesuaikan gerakan dengan kondisi pengguna, deteksi gerakan secara real-time, pelacakan tingkat nyeri, hingga visualisasi progres dalam bentuk grafik. Inovasi utamanya terletak pada sistem rekomendasi yang responsif terhadap kondisi pengguna serta kemampuan aplikasi memberikan umpan balik terhadap performa latihan, sehingga terapi menjadi lebih personal, terarah, dan terukur.

3. Dampak Pemanfaatan Aplikasi Spinecare

Dengan adanya SpineCare, pengguna yang mengalami nyeri punggung, (khususnya dari kalangan muda dan produktif) dapat melakukan terapi secara mandiri dengan panduan yang jelas dan mudah diakses. Aplikasi ini membantu menjembatani keterbatasan akses ke fasilitas medis dan meningkatkan kesadaran akan pentingnya kesehatan punggung bawah. Selain itu, penggunaan teknologi seperti ini berpotensi mengurangi beban fasilitas kesehatan untuk kasus ringan, sekaligus memperluas penerapan digital health di Indonesia. Harapannya, Spinecare dapat menjadi solusi jangka panjang yang tidak hanya meredakan nyeri, tetapi juga mendorong perubahan gaya hidup yang lebih sehat.

PEMBAHASAN APLIKASI

1. INSTALASI DAN KEBUTUHAN SISTEM

Sebelum memulai, pastikan perangkat Anda memenuhi persyaratan berikut untuk dapat menjalankan aplikasi SpineCare dengan optimal.

1.1. Persyaratan Sistem

- Sistem Operasi : Android 8.0 atau versi lebih baru
- Perangkat Keras : Kamera depan yang aktif
- Koneksi : Koneksi internet yang stabil

1.2. Cara Instalasi Aplikasi

1) Pastikan Anda telah mengaktifkan izin "Instalasi dari sumber tidak dikenal" pada perangkat Android Anda
   (Pengaturan > Keamanan > aktifkan Unknown Sources atau Sumber Tidak Dikenal).
2) Unduh file Spinecare.apk melalui tautan yang telah disediakan.
3) Setelah selesai diunduh, buka file Spinecare.apk dari folder unduhan Anda.
4) Klik “Install” dan tunggu hingga proses instalasi selesai.
5) Aplikasi Spinecare siap digunakan.

2. DESKRIPSI FUNGSIONAL APLIKASI SPINECARE

Aplikasi Spinecare dirancang untuk membantu mengurangi nyeri punggung bawah secara mandiri melalui panduan gerakan terapi yang disesuaikan dengan kondisi dan keluhan pengguna. Aplikasi ini dilengkapi fitur rekomendasi gerakan, deteksi ketepatan latihan, dan pemantauan progres latihan agar proses pemulihan lebih tepat dan terukur.

3. PENJELASAN DETAIL FITUR

Untuk mendukung fungsi utama tersebut dan memastikan pengalaman pengguna yang lengkap dan aman, SpineCare menyediakan serangkaian fitur utama berikut sebelum Anda masuk ke langkah-langkah penggunaannya:

- Pendaftaran dan Verifikasi Akun
  Proses pembuatan akun baru yang aman melalui verifikasi email untuk memastikan validitas pengguna.

- Login Pengguna
  Proses pembuatan akun baru yang aman melalui verifikasi email untuk memastikan validitas pengguna.

- Rekomendasi Latihan Personal
  Sistem untuk mendapatkan program latihan yang disesuaikan setelah menjawab kuesioner kondisi nyeri.

- Sesi Terapi dengan Deteksi Gerakan
  Fitur inti di mana aplikasi secara real-time mendeteksi dan memberikan umpan balik atas ketepatan gerakan terapi yang dilakukan pengguna.

- Analisis Performa Latihan
  Halaman statistik untuk memantau perkembangan tingkat nyeri melalui grafik dan mendapatkan umpan balik otomatis dari sistem.

4. PENDAFTARAN DAN LOGIN AKUN

Untuk mengakses fitur-fitur SpineCare, Anda perlu memiliki akun.

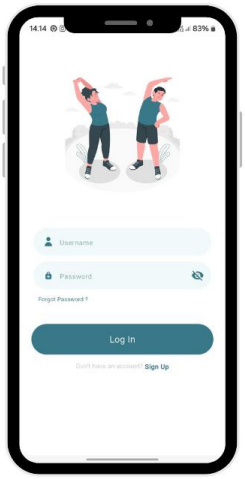
4.1. Bagi Pengguna Baru (Registrasi)

Bagi Pengguna yang belum memiliki akun, setelah intro aplikasi akan langsung diarahkan ke halaman registrasi untuk melakukan proses pendaftaran.

1) Lengkapi formulir pendaftaran dengan mengisi data berikut:
   - username
   - alamat email
   - nomor hp
   - password
   - konfirmasi password
2) Setelah semua data terisi, klik tombol “Sign Up” untuk membuat akun

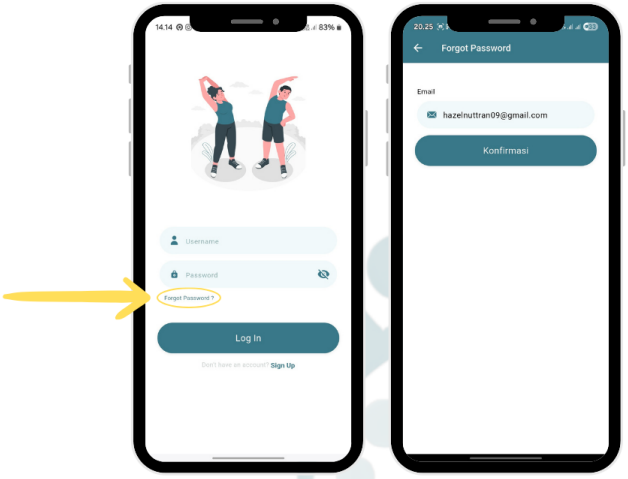
4.2. Bagi Pengguna Terdaftar (Login)

1) Jika sudah memiliki akun, masukkan Email dan Password pada halaman login.
2) Klik tombol “Log In” untuk masuk ke aplikasi.

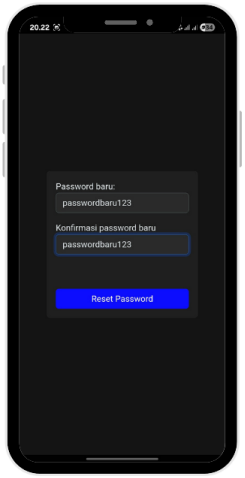


4.3. Mengatasi Lupa Password

1) Jika Anda lupa password saat akan login, klik tautan "Forgot Password?" yang terdapat pada halaman login.
2) Anda akan diarahkan ke halaman "Forgot Password". Masukkan alamat email yang terdaftar pada akun Anda.
3) Klik tombol "Konfirmasi" dan ikuti instruksi pemulihan yang dikirimkan ke email Anda.



4) Di dalam email tersebut, klik tautan (link) yang disediakan. Anda akan diarahkan ke halaman web khusus untuk mengganti password.
5) Pada halaman tersebut, isi kolom "Password baru" dan “Konfirmasi password baru”.
6) Klik tombol "Reset Password" untuk menyimpan.
7) Sekarang Anda dapat kembali ke aplikasi SpineCare dan login menggunakan password baru Anda.



5. MEMULAI TERAPI DENGAN SPINECARE

Setelah berhasil login, ikuti langkah-langkah berikut untuk memulai sesi terapi Anda.

5.1. Persetujuan Pengguna (Penting!)

Sebelum mengakses fitur utama, Anda akan diarahkan ke halaman Persetujuan Pengguna. Anda wajib membaca dan menyetujui syarat serta ketentuan yang berlaku.

Kondisi Darurat: Aplikasi ini tidak ditunjukan untuk pengguna dengan gejala darurat berikut:

- Kehilangan kendali buang air.
- Mati rasa di area selangkangan.
- Kehilangan kendali buang air.
- Nyeri pinggang yang disertai kelemahan pada kaki.

Mekanisme Persetujuan:

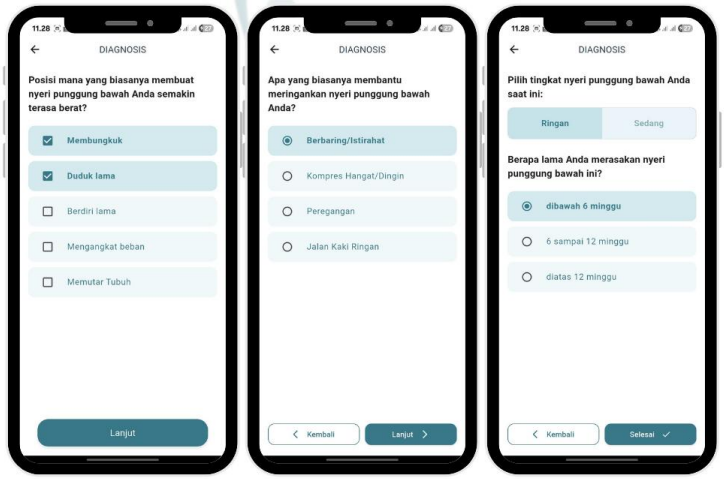
- Jika Anda tidak memiliki gejala darurat, pilih "Tidak" pada pertanyaan "Memiliki gejala darurat" untuk melanjutkan.
- Jika Anda memiliki salah satu gejala darurat di atas, Anda tidak diperkenankan melanjutkan sesi terapi dan hanya dapat mengakses fitur Artikel Kesehatan. Segera konsultasikan kondisi Anda dengan dokter.

5.2. Personalisasi Program Latihan

1) Pada halaman utama (dashboard), klik tombol **"Dapatkan Rekomendasi"** untuk mendapatkan rekomendasi gerakan yang sesuai dengan kondisi Anda.

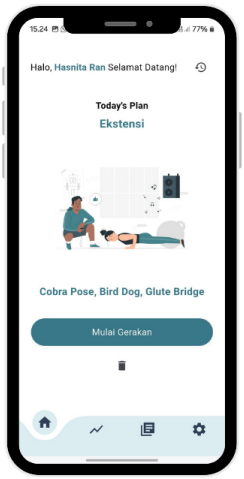
2) Selanjutnya, jawab serangkaian pertanyaan personalisasi sesuai dengan apa yang Anda rasakan. Pertanyaan meliputi:

- Posisi yang memperberat nyeri.
- Aktivitas yang meringankan nyeri.
- Durasi Anda mengalami nyeri.
- Tingkat nyeri Anda saat ini.



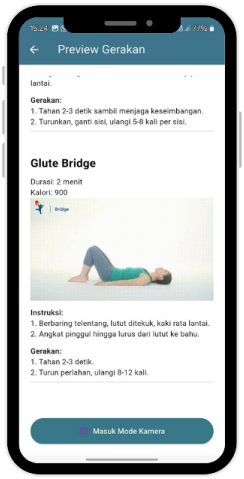
3) Setelah semua pertanyaan terjawab, klik tombol "Selesai" untuk melihat hasil rekomendasi gerakan.

5.3. Melakukan Gerakan Terapi



1) Setelah personalisasi program, aplikasi akan menampilkan rencana latihan (Contoh: Ekstensi) beserta daftar gerakan yang direkomendasikan (Contoh: Cobra Pose, Bird Dog, Glute Bridge).

2) Anda dapat melihat instruksi detail setiap gerakan terlebih dahulu. Klik pada Mulai Gerakan untuk melihat preview sesi latihan.

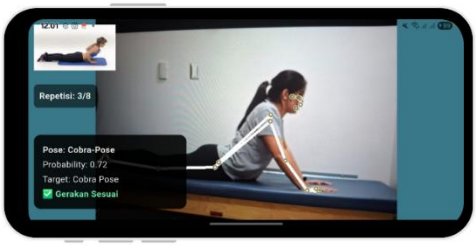


Pada preview gerakan anda bisa melihat informasi setiap daftar gerakan yang perlu Anda lakukan diantaranya:

- Instruksi: Cara melakukan posisi awal
- Gerakan: Urutan gerakan yang harus dilakukan, durasi menahan posisi, dan jumlah repetisi.

3) Jika sudah siap, klik "Masuk Mode Kamera" untuk memulai sesi latihan.

4) Posisikan diri Anda di depan kamera dan ikuti video instruksi yang ditampilkan di pojok kiri atas layar. Sistem akan mendeteksi dan memberikan umpan balik secara real-time mengenai:



- Gerakan yang terdeteksi.
- Tingkat keakuratan gerakan (Probability)
- Kesesuaian gerakan Anda dengan target

6. MEMANTAU PROGRES DAN PERFORMA

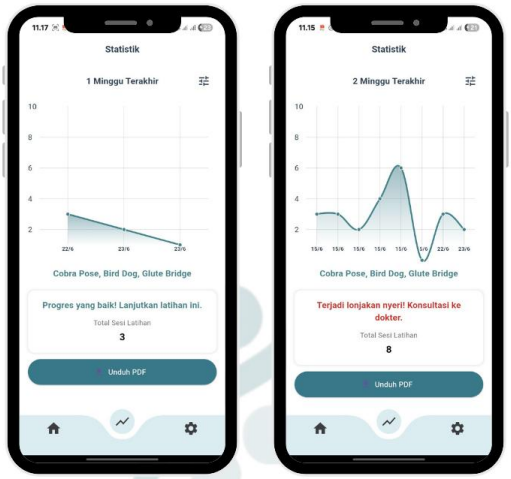
Aplikasi SpineCare membantu Anda melacak perkembangan kondisi dari waktu ke waktu. Untuk mengaksesnya, klik ikon grafik (📈) yang berada di tengah bawah pada menu navigasi.

6.1 Input Skala Nyeri

Setelah menyelesaikan satu rangkaian gerakan terapi, sebuah pop-up akan muncul. Anda diwajibkan untuk menggeser slider dan memilih tingkat nyeri yang Anda rasakan setelah berlatih, lalu klik "Kirim".

6.2 Halaman Performa Nyeri (Statistik)

1) Setelah mengirim input nyeri, Anda akan otomatis diarahkan ke halaman Statistik untuk melihat grafik performa nyeri Anda. Anda juga bisa mengakses halaman ini kapan saja dengan mengklik ikon grafik di menu navigasi bawah.



2) Pada halaman Statistik, grafik performa akan menampilkan perkembangan tingkat nyeri Anda. Berdasarkan data ini, sistem akan memberikan umpan balik otomatis untuk memandu langkah Anda selanjutnya.

Rekomendasi Positif (Lanjutkan Latihan)

Anda akan menerima pesan untuk melanjutkan latihan jika progres Anda dinilai aman dan positif:

- Jika terjadi penurunan nyeri: Sistem akan menampilkan pesan seperti "Progres yang baik! Lanjutkan latihan ini." Ini menunjukkan bahwa program latihan memberikan dampak positif.
- Jika nyeri cenderung stabil: Anda akan melihat pesan seperti "Progres Anda stabil. Tetap semangat!", yang menandakan tidak ada perburukan kondisi dan latihan dapat dilanjutkan dengan aman.

Peringatan & Saran Konsultasi

Untuk menjaga keamanan Anda, sistem akan memberikan peringatan dan menyarankan Anda untuk berkonsultasi dengan dokter jika mendeteksi salah satu dari kondisi berikut:

- Nyeri Sangat Tinggi: Jika Anda melaporkan tingkat nyeri di atas ambang batas aman (misalnya, di atas 7), sistem akan memberikan peringatan "Nyeri sangat tinggi! Segera konsultasi ke dokter."
- Lonjakan Nyeri Drastis: Jika tingkat nyeri Anda meningkat tajam dalam satu sesi (misalnya, naik 3 poin atau lebih), Anda akan diperingatkan dengan pesan "Terjadi lonjakan nyeri! Konsultasi ke dokter."
- Tren Nyeri yang Terus Meningkat: Jika setelah beberapa sesi latihan grafik menunjukkan nyeri Anda secara konsisten cenderung meningkat, sistem akan menyarankan "Nyeri cenderung meningkat. Konsultasi ke dokter."
- Tidak Ada Kemajuan (Stagnan): Jika setelah cukup banyak sesi latihan (misalnya lebih dari 8 sesi) tidak ada perubahan yang signifikan pada tingkat nyeri Anda, sistem akan memberikan masukan "Tidak ada kemajuan signifikan. Sebaiknya konsultasi."

Umpan balik ini dirancang untuk memastikan Anda berlatih secara efektif dan aman, serta mendapatkan penanganan medis yang tepat ketika dibutuhkan.

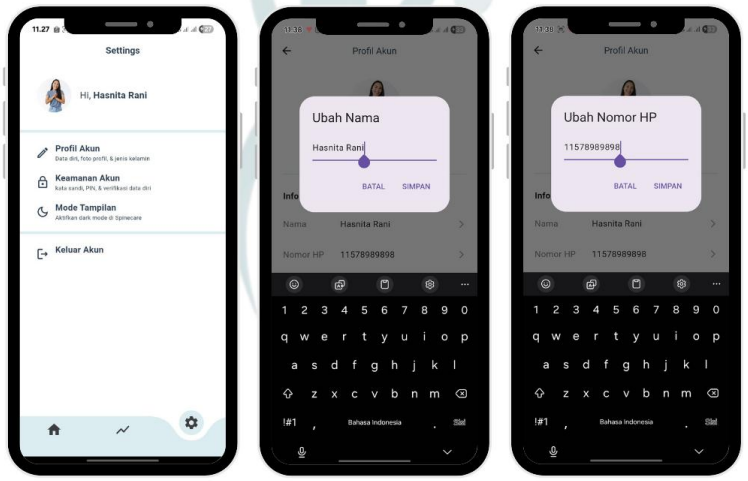
7. PENGATURAN DAN MANAJEMEN AKUN

Menu Pengaturan memungkinkan Anda untuk mengelola profil, mengubah tampilan aplikasi, dan menjaga keamanan akun Anda. Untuk mengaksesnya, klik ikon gerigi (⚙) yang berada di pojok kanan bawah pada menu navigasi.

7.1 Mengelola Profil Pengguna

Pada menu Pengaturan, pilih "Profil Akun" untuk melihat dan mengelola informasi pribadi Anda. Di halaman ini, Anda dapat:

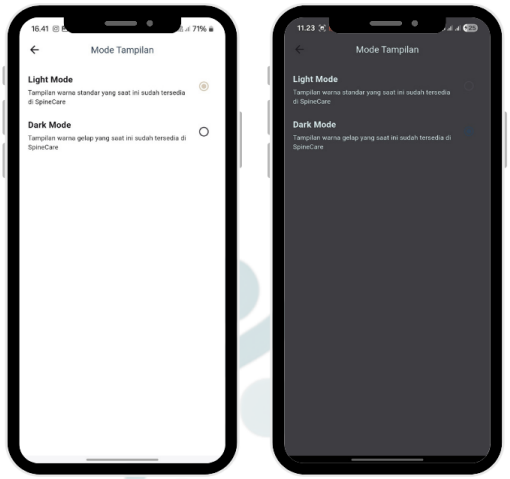
- Melihat informasi profil seperti Nama Lengkap, Nomor HP, Jenis Kelamin, dan Tanggal Lahir.
- Mengubah foto profil Anda dengan memilih opsi "Ubah Foto Profil".



7.2 Mengubah Tampilan Aplikasi (Tema)

SpineCare menyediakan dua mode tampilan untuk kenyamanan visual Anda.

1) Masuk ke menu Pengaturan, lalu pilih "Mode Tampilan".
2) Pilih antara "Light Mode" (mode terang) atau "Dark Mode" (mode gelap) sesuai preferensi Anda.



7.3 Keamanan Akun

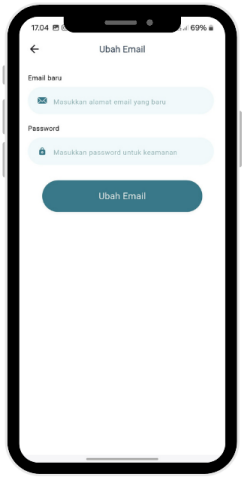
Jaga keamanan akun Anda dengan memperbarui email atau password secara berkala melalui menu "Keamanan Akun".

- Mengubah Alamat Email:
  1) Di dalam menu "Keamanan Akun", pilih "Ubah Email".
  2) Masukkan alamat email baru dan password Anda saat ini untuk konfirmasi, lalu simpan perubahan.

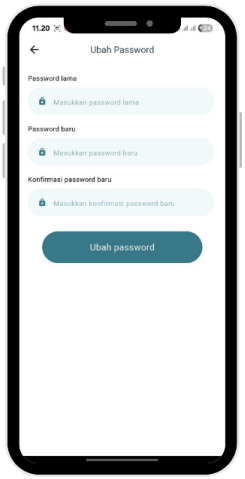
- Mengubah Password:
  1) Di dalam menu "Keamanan Akun", pilih "Ubah Kata Sandi".
  2) Masukkan password lama Anda, kemudian masukkan password baru dan konfirmasikan sekali lagi untuk menyelesaikan proses.

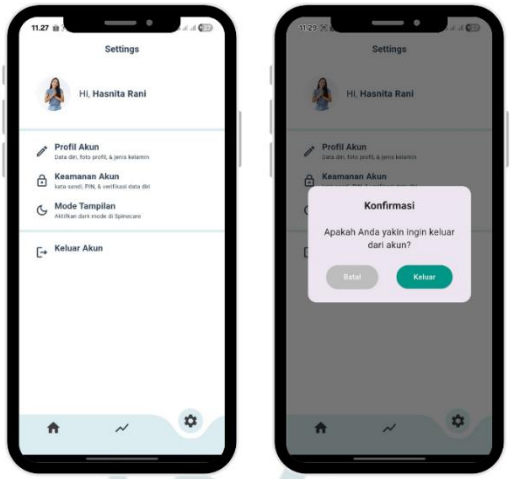
Jaga keamanan akun Anda dengan memperbarui email atau password secara berkala melalui menu "Keamanan Akun".

- Mengubah Alamat Email:
  1) Di dalam menu "Keamanan Akun", pilih "Ubah Email".
  2) Masukkan alamat email baru dan password Anda saat ini untuk konfirmasi, lalu simpan perubahan.



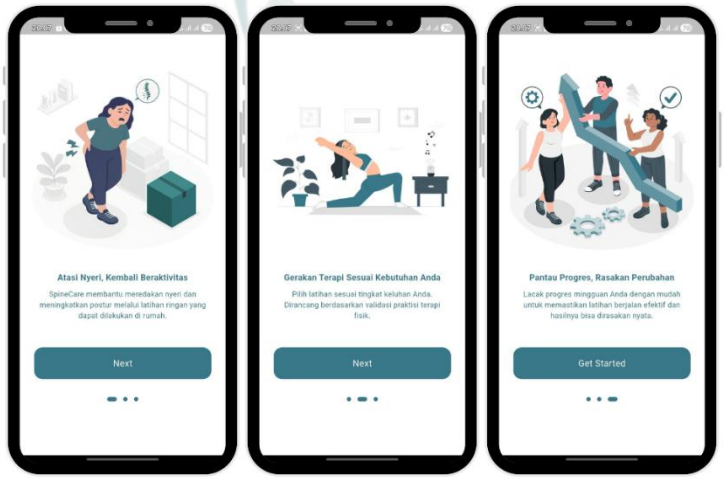
- Mengubah Password:
  1) Di dalam menu "Keamanan Akun", pilih "Ubah Kata Sandi".
  2) Masukkan password lama Anda, kemudian masukkan password baru dan konfirmasikan sekali lagi untuk menyelesaikan proses.





- Logout Akun

  Klik menu “Keluar Akun” untuk keluar dari aplikasi dan mengakhiri sesi penggunaan akun Anda.



Lampiran 6 Dokumen Teknis Aplikasi

Pendahuluan

1. Latar Belakang

Nyeri punggung bawah *(low back pain)* merupakan salah satu masalah muskuloskeletal paling umum di dunia dan menjadi penyebab utama disabilitas global. WHO memperkirakan angka kasus *low back pain* akan terus meningkat. Kondisi ini tidak hanya dialami lansia, tetapi juga menyerang kelompok usia produktif.

Faktor gaya hidup modern seperti duduk terlalu lama, kurang aktivitas fisik, dan penggunaan gawai dengan postur tidak ergonomis menjadi penyebab utama. Di Indonesia, mahasiswa dan pekerja muda termasuk kelompok yang rentan mengalami keluhan ini akibat durasi duduk yang panjang dan posisi tubuh yang tidak ideal selama aktivitas sehari-hari.

Penanganan nyeri punggung bawah umumnya dilakukan melalui terapi fisik atau pengobatan, namun akses ke layanan kesehatan seringkali terbatas. Terapi mandiri menjadi alternatif yang potensial, tetapi banyak individu mengalami kesulitan dalam memilih gerakan yang tepat, menjaga konsistensi, atau melakukan latihan dengan benar tanpa pendampingan.

Untuk menjawab tantangan tersebut, dikembangkanlah aplikasi SpineCare—sebuah solusi digital yang menyediakan panduan latihan personal, deteksi gerakan otomatis, pelacakan nyeri secara visual, serta sistem rekomendasi lanjutan. Aplikasi ini diharapkan dapat mendukung terapi mandiri secara lebih aman, efisien, dan terstruktur, sekaligus mendorong penerapan teknologi dalam layanan kesehatan preventif di Indonesia.

2. Tujuan Pembuatan Dokumen

Dokumen ini disusun untuk menjelaskan latar belakang, rancangan, serta fungsi utama dari aplikasi Spinecare sebagai solusi terapi mandiri untuk nyeri punggung bawah. Penulisan ini juga bertujuan untuk mendokumentasikan proses pengembangan aplikasi, mulai dari identifikasi masalah, perancangan sistem, hingga fitur-fitur yang diimplementasikan. Selain itu, dokumen ini menjadi bentuk pertanggungjawaban atas proses pengembangan dan diharapkan dapat menjadi referensi bagi pengembangan lanjutan atau penelitian serupa di bidang teknologi kesehatan.

3. Nilai Inovasi

SpineCare menawarkan pendekatan baru dalam terapi mandiri berbasis teknologi. Tidak hanya menyajikan video latihan seperti aplikasi sejenis, Spinecare dilengkapi fitur asesmen awal untuk menyesuaikan gerakan dengan kondisi pengguna, deteksi gerakan secara real-time, pelacakan tingkat nyeri, hingga visualisasi progres dalam bentuk grafik. Inovasi utamanya terletak pada sistem rekomendasi yang responsif terhadap kondisi pengguna serta kemampuan aplikasi memberikan umpan balik terhadap performa latihan, sehingga terapi menjadi lebih personal, terarah, dan terukur.

4. Dampak Pemanfaatan Aplikasi Spinecare

Dengan adanya SpineCare, pengguna yang mengalami nyeri punggung, (khususnya dari kalangan muda dan produktif) dapat melakukan terapi secara mandiri dengan panduan yang jelas dan mudah diakses. Aplikasi ini membantu menjembatani keterbatasan akses ke fasilitas medis dan meningkatkan kesadaran akan pentingnya postur tubuh yang benar. Selain itu, penggunaan teknologi seperti ini berpotensi mengurangi beban fasilitas kesehatan untuk kasus ringan, sekaligus memperluas penerapan digital health di Indonesia. Harapannya, Spinecare dapat menjadi solusi jangka panjang yang tidak hanya meredakan nyeri, tetapi juga mendorong perubahan gaya hidup yang lebih sehat.

5. Spesifikasi Teknis

Spesifikasi Teknis dari Aplikasi Spinecare meliputi:

- Manual Pengguna
- Source Code

Berikut uraian spesifikasi untuk pembangunan aplikasi:

- Python (Flask)
- Dart (Flutter)
- Google Colab
- Visual Studio Code
- Web Browser
- MongoDB

1. Register

```python
@bp.route('/register', methods=['POST'])
@api_key_required
def register():
    data = request.get_json()
    return register_user(data)
```

Endpoint menerima permintaan POST pada url /auth/register dan diproteksi dengan dekorator @api_key_required. Fungsi register() mengambil json dari body permintaan menggunakan request.get_json(). Data diteruskan ke register_user(data).

```python
def register_user(data):
    ...
    # Confirmation Password Check
    ...
    # Check Email Already Registered
    user = db.db.users.find_one({"email": email})
    if user:
        return {
            'message': 'Email ini telah terdaftar sebelumnya, gunakan email yang lain!'
        }, 400

    hashed_password = PasswordHasher().hash(password)

    try:
        user = {
            "fullname": fullname,
            "email": email,
            "no_hp": no_hp,
            "password": hashed_password,
            "photo": default_photo,
            "gender": gender,
            "is_verify": False,
            "created_at": datetime_now,
            "updated_at": datetime_now,
        }

       ...

        result = db.db.users.insert_one(user)
        result = db.db.token.insert_one(token)

        ...
        mail.send(msg)

        return {
            "message": "Berhasil mendaftar, cek email untuk verifikasi"
        }, 201
```

Fungsi register_user(data) akan menerima input berupa data seperti fullname, email, no_hp, gender, password. Kemudian sistem melakukan pengecekan apakah email yang dimasukan sudah terdaftar di (db.db.users). Jika sudah terdaftar, sistem akan menghentikan proses dan memberikan respon yang sesuai, Namun jika email valid, password akan di hash menggunakan Argon2 demi keamanan. Data pengguna kemudian disiapkan dalam bentuk dictionary, termasuk is_verify dengan default False.

Setelah data disimpan dalam koleksi users, sistem menghasilkan token verifikasi secrets.token_urlsafe() kemudian menyimpannya ke dalam koleksi token. Token disertakan ke link verifikasi email yang dibuat dengan fungsi url_for(), dan dikirim ke alamat pengguna melalui modul Flask-Mail. Template HTML email verifikasi dirender menggunakan render_template("verify-email.html", data=data).

Jika semua langkah berhasil sistem memberikan respon 201 (sukses).

2. Verifikasi Akun

```python
@bp.route('/verify_account/<token>', methods=['GET'])
def verify_account(token):
    return verif_user(token)
```

Endpoint /verify_account/<token> menerima parameter token yang dimasukan ke dalam url kemudian diteruskan ke fungsi verif_user(token) untuk diproses lebih lanjut. metode GET akan melakukan verifikasi saat pengguna klik link verifikasi di email mereka.

```python
def verif_user(token):
    try:
        token = db.db.token.find_one({"token": token})
        if not token:
            response = make_response(render_template('response.html', success=True, message='Account has ben verified'), 400)
            response.headers['Content-Type'] = 'text/html'
            return response

        email = token["email"]
        user = db.db.users.find_one({"email": email})

        if user["is_verify"] == True:
            response = make_response(render_template('response.html', success=False, message='Account has been verified'), 400)
            response.headers['Content-Type'] = 'text/html'
            return response

        update_data = {"is_verify" : True}
        db.db.users.update_one({"email": email}, {"$set": update_data})

        response = make_response(render_template('response.html', success=True, message='Account has ben verified'), 400)
        response.headers['Content-Type'] = 'text/html'

        db.db.token.delete_one({"token": token})

        return response

    except Exception as e:
        return {"message": f"Error {e}"}, 500
```

Proses verifikasi dimulai dengan mencari token yang diterima pada koleksi token di db.db.token.find_one ({"token": token}). Token disini berisi email pengguna yang perlu diverifikasi. Jika token tidak ditemukan, sistem mengembalikan respons HTML dengan status 400 (terverifikasi). Jika token tidak ditemukan, aplikasi melanjutkan untuk memeriksa status verifikasi pengguna menggunakan db.db.users.find_one ({"email": email}). Jika sudah terverifikasi ("is_verify" == True), maka sistem mengembalikan respons HTML bahwa akun telah terverifikasi sebelumnya.
Jika akun belum terverifikasi, sistem akan memperbarui status pengguna menjadi True di database melalui perintah update_one().
Setelah akun berhasil diverifikasi, token yang digunakan akan dihapus dari koleksi token untuk mencegah penggunaan berulang. Respons HTML yang menyatakan akun berhasil diverifikasi dikembalikan ke pengguna dengan statis 200.

3. Login

```python
@bp.route('/login', methods=['POST'])
@api_key_required
def login():
    data = request.headers.get('Authorization')
    return login_user(data)
```

Proses login dimulai dengan penerimaan permintaan yang menyertakan informasi otentikasi pengguna dalam header Authorization dengan format Basic Auth. diikuti dengan data login yang telah base64 encoded dalam format Basic <base64encoded(email:password)>. Permintaan yang valid akan diteruskan ke fungsi login_user(data) untuk diproses lebih lanjut.

```python
def login_user(data):
    base64Str = data[6:] # hapus "Basic" string

    #Mulai Base64 Decode
    base64Bytes = base64Str.encode('ascii')
    messageBytes = base64.b64decode(base64Bytes)
    pair = messageBytes.decode('ascii')
    #Akhir Base64 Decode

    email, password = pair.split(":")

    user = db.db.users.find_one({"email": email})

    if not user:
        return {
            "message": "Pengguna tidak ditemukan, silahkan register terlebih dahulu"
        }, 400

    if user["is_verify"] == False:
        return {
            "message": "Pengguna berlum terverifikasi, silahkan cek email"
        }

    try:
        if PasswordHasher().verify(user["password"], password):
            payload = {
                'id': str(user["_id"]),
                'email': user["email"],
                'fullname': user["fullname"]
            }

            token = create_access_token(identity=email)

            return {'message': 'Berhasil login',
                        'user': payload,
                        'token': token
                        }, 200
        else:
            return { 'message': 'Password salah' }, 401
    except VerifyMismatchError:
        return { 'message': 'Password tidak sesuai' }, 401
```

Fungsi login_user(data) mengambil string yang dikirimkan dalam header Authorization dan menghapus kata “Basic” menggunakan data[6:]. Selanjutnya, string yang tersisa (base64 encoded) akan di-decode menggunakan base64.b64decode(), kemudian hasil decode tersebut diubah menjadi pasangan email dan password menggunakan metode split(“:”). Aplikasi akan mencari pengguna menggunakan db.db.users.find_one({”email”: email}). Jika tidak ditemukan sistem mengembalikan respons 400 (“pengguna tidak ditemukan, silahkan registrasi”). Sistem akan memeriksa field is_verify, jika (is_verify == False) sistem mengirim respons “Pengguna belum terverifikasi, cek email”.

Selanjutnya aplikasi memverifikasi kecocokan password menggunakan PasswordHasher().verify(user[“password”], password). Jika password tidak sesuai sistem mengembalikan pesan kesalahan (401). Jika verifikasi password berhasil, aplikasi membuat JWT (JSON Web Token) untuk sesi pengguna menggunakan create_access_token(identity=email). Token ini digunakan untuk autentikasi sesi berikutnya yang mengandung informasi seperti ID, email dan nama lengkap.

Jika login berhasil, sistem mengembalikan respons 200 (sukses), user dan token JWT dapat digunakan untuk otentikasi lebih lanjut.

4. Get Profil

```python
@bp.route('/profile', methods=['GET'])
@jwt_required()
@api_key_required
def profile():
    current_user = get_jwt_identity()
    return get_profile(current_user)
```

Endpoint GET /user/profile mengharuskan pengguna melakukan otentikasi dengan token JWT yang valid. Selain itu memastikan permintaan yang masuk bersumber dari terauntetikasi menggunakan API key.

Fungsi get_jwt_identity() mendapatkan email pengguna yang sedang login berdasarkan token JWT yang mereka miliki. Kemudian get_profile() akan mencari data pengguna berdasarkan email yang diperoleh dari token JWT di database MongoDB.

```python
def get_profile(email):
    try:
        user = db.db.users.find_one({"email": email})

        if not user:
            return {
                'message': 'Pengguna tidak ditemukan'
            }, 404

        return {
            'message': 'success',
            'data': {
                'id': str(user["_id"]),
                'fullname': user["fullname"],
                'email': user["email"],
                'no_hp': user["no_hp"],
                'photo': user["photo"],
            }
        }, 200

    except Exception as e:
        return {'message' : f"Error {e}"}, 500
```

Fungsi get_profile(email) mengambil data profil pengguna berdasarkan email yang diberikan. Fungsi ini akan mencari data pengguna di koleksi users pada database MongoDB. Jika pengguna tidak ditemukan, mengembalikan respons 404 (“Pengguna tidak ditemukan”). Jika pengguna ditemukan, mengembalikan informasi profil pengguna, seperti ID, nama lengkap, email, nomor HP dan foto.

Jika ditemukan maka respons yang dihasilkan 200 OK dengan data pengguna. Jika tidak ditemukan responsnya 404 Not Found dengan pesan kesalahan. Jika terjadi kesalahan lain responnya adalah 500 Internal Server Error.

5. Forgot Password

```python
@bp.route('/forgot-password', methods=['POST'])
@api_key_required
def forgot_password_endpoint():
    data = request.get_json()
    print(data)
    return forgot_password_user(data)
```

Pengguna mengirimkan permintaan POST dengan email mereka di body request. Fungsi forgot_password_user() akan memeriksa apakah email terdaftar, jika valid, akan mengirimkan email verifikasi.

```python
def forgot_password_user(data):
    try:
        email = data["email"]

        user = db.db.users.find_one({"email": email})

        if not user:
            return {
                "message": "Email tidak terdaftar"
            }, 404

        verification_token = secrets.token_urlsafe(32)

        token = {
            "email": email,
            "token": verification_token
        }

        db.db.token.insert_one(token)

        confirmation_url = url_for('user.reset_password_view', token=verification_token, _external=True)

        msg = Message(subject="Reser Your Password - SpineMotion", sender="spinemotionapp@gmail.com", recipients=[email])
        msg.html = render_template("reset-password.html", url=confirmation_url)

        mail.send(msg)

        return {'message' : "Berhasil meminta reset password, silahkan cek email"}, 200

    except Exception as e:
        return {'message' : f"Error {e}"}, 500
```

Fungsi ini mengambil email dari data JSON yang dikirimkan oleh pengguna dalam body permintaan. Kemudian mencari pengguna di database berdasarkan email menggunakan db.db.users.findone({"email": email}). Jika pengguna tidak ditemukan, maka sistem akan mengembalikan respons status 404 Not Found dan pesan bahwa email tidak terdaftar. Jika pengguna ditemukan, fungsi menghasilkan token verfikasi yang bersifat unik menggunakan secrets.token_urlsafe(32). Token ini digunakan untuk memastikan bahwa proses reset password hanya berlaku untuk pengguna yang valid. Token ini kemudian disimpan di database ke dalam koleksi token, dengan email yang terkait. Fungsi selanjutnya membuat link konfirmasi (link untuk reset password) menggunakan url_for(), yang mengarahkan pengguna ke halaman reset passeord dengan token tersebut. Email dikirmkan ke pengguna menggunakan Flask-Mail. Template HTML untuk email di-render dengan url konfirmasi dan kemudian dikirimkan ke alamat email pengguna. Setelah email berhasil dikirim, fugsi mengembalikan respons 200 OK (sukses), memberi tahu pengguna kalau permintaan reset password telah berhasil dan mereka harus memeriksa email mereka. Jika terjadi kesalahan, fungsi mengembalikan respons 500 Internal Server Error.

6. Reset Password

```python
@bp.route('/reset-password-view/<token>', methods=['GET'])
def reset_password_view(token):
    return reset_password_view_user(token)
```

Fungsi mencari token yang diterima dalam koleksi token di database. Jika token valid, formulir reset password akan ditampilkan.

```python
def reset_password_view_user(token):
    try:
        token = db.db.token.find_one({"token": token})
        if not token:
            return {
                'message':'Token not found'
            }, 404

        email = token["email"]
        user = db.db.users.find_one({"email": email})
        if not user:
            return {
                "message": "User not found"
            }, 404

        response = make_response(render_template('form-reset-password.html', email=user["email"]), 200)
        response.headers["Content-Type"] = "text/html"
        return response
    except Exception as e:
        return {
            "message": f"Error {e}"
        }, 500
```

Pengguna akses link reset password yang berisi token. Jika token valid, fungsi reset_password_view_user(token) akan merender formulir reset password. Jika token tidak valid atau tidak ditemukan, akan dikembalikan status 404 Not Found.

```python
@bp.route('/reset-password', methods=['POST'])
def reset_password():
    password = request.form.get("password")
    confirm_password = request.form.get("confirmPassword")
    email = request.form.get("email")
    return reset_password_user(password, confirm_password, email)
```

Pengguna mengirimkan permintaan POST dengan data password baru dan konfirmasi password. Fungsi reset_password_user() memverifikasi kecocokan password dan memperbarui password di database jika valid. Jika berhasil, password baru akan di-hash dan disimpan, serta token reset akan dihapus. Jika ada kesalahan, pengembalian status 400 Bad Request atau 500 Internal Server Error dilakukan.

```python
def reset_password_user(password, confirm_password, email):
    try:
        user = db.db.users.find_one({"email": email})

        if not user:
            return {
                "message": "Pengguna tidak ditemukan"
            }, 404

        if password != confirm_password:
            return {
                "message": "Password tidak sesuai"
            }

        new_password = PasswordHasher().hash(password)

        update_password = {
            "password": new_password
        }

        try:
            db.db.users.update_one({"email": email}, {"$set": update_password})
            db.db.token.delete_one({"email": email})
            response = make_response(render_template('response.html', success=True, message='Password has been reset successfully'), 200)
            response.headers['Content-Type'] = 'text/html'
            return response

        except:
            response = make_response(render_template('response.html', success=False, message='Reset password failed'), 400)
            response.headers['Content-Type'] = 'text/html'
            return response
    except Exception as e:
        return {
            "message": f"Error {e}"
        }, 500
```

Fungsi reset_password_user() mencari pengguna di database menggunakan email yang diterima dalam parameter. Jika pengguna tidak ditemukan di database, sistem akan mengembalikan status 404 Not Found (pesan kesalahan).
Jika password dan confirm_password sesuai, password baru akan di-hash oleh PasswordHasher() dari argon2 untuk menjaga keamanan. Setelah password di-hash, password baru akan disimpan di database, menggantikan password yang lama. token yang digunakan untuk reset password juga akan dihapus dari koleksi token, untuk mencegah penggunaan token yang sama lagi. Jika semua langkah berhasil, respons 200 (sukses) dikembalikan dan jika ada kesalahan dalam pembaruan data atau penghapusan token, maka respons 400 Bad Request dikembalikan. Namun jika terjadi kesalahan selama proses apapun (misal kesalahan koneksi database atau kesalahan lainnya), sistem akan mengembalikan 500 Internal Server Error.

7. Change Password

```python
@bp.route('/change-password', methods=['PUT'])
@jwt_required()
@api_key_required
def change_password():
    data = request.get_json()
    return change_password_user(data)
```

Pengguna mengirimkan permintaan PUT ke /user/change-password dengan data JSON yang berisi oldPassword, password, dan confirmPassword. Fungsi change_password(data) dipanggil untuk memproses data dan memperbarui password di database.

```python
def change_password_user(data):
    old_password = data["oldPassword"]
    password = data["password"]
    confirm_password = data["confirmPassword"]
    email = get_jwt_identity()

    user = db.db.users.find_one({"email": email})
    try:
        if PasswordHasher().verify(user["password"], old_password):
            if password != confirm_password:
                return {
                    "message": "Password baru tidak cocok, cek kembali password nya!"
                }, 401

            # Hash password baru
            new_password_hashed = PasswordHasher().hash(password)

            # simpan password baru
            update_password = {
                "password": new_password_hashed
            }

            db.db.users.update_one({"email": email}, {"$set": update_password})

            return {
                "message": "Berhasil mengubah password"
            }, 200
    except VerifyMismatchError:
        return {
            "message": "Password lama tidak sesuai"
        }, 401
    return "change pws"
```

Fungsi akan mengambil password lama, password baru, dan konfirmasi password dari data JSON yang dikirimkan oleh pengguna. PasswordHasher().verify() memverifikasi hash password yang tersimpan di database dengan data yang dimasukan.

Jika old_password valid, fungsi memeriksa apakah password baru yang dimasukan sesuai dengan konfirmasi password. Jika tidak sesuai, fungsi mengembalikan respons 401 Unauthotized (pesan kesalahan). Jika password baru valid, password tersebut di-hash menggunakan PasswordHasher() dan disimpan di database. Fungsi memperbarui password di database dengan db.db.users.update_one(). Jika password berhasil diperbarui fungsi mnegembalikan respons 200 OK (sukses). Jika ada kesalahan dalam porses (misal password lama tidak valid) maka fungsi mengembalikan respons 401 Unauthorized,

```python
@bp.route('/request-change-email', methods=['POST'])
@jwt_required()
@api_key_required
def change_email():
    data = request.get_json()
    return request_change_email_service(data)
```

Pengguna mengirimkan POST dengan data JSON berisi new_email dan password. Fungsi request_change_email_service(data) dipanggil untuk memverifikasi password dan menghasilkan OTP untuk email baru.

```python
def request_change_email_service(data):
    email = get_jwt_identity()
    new_email = data["new_email"]
    password = data["password"]

    user = db.db.users.find_one({"email": email})

    if not user:
        return {
            "message": "Pengguna tidak ditemukan, silahkan register terlebih dahulu"
        }, 400

    try:
        if PasswordHasher().verify(user["password"], password):

            otp = random.randint(100000, 999999)

            created_at = datetime.now().strftime("%Y%m%d%H%M%S")

            data = {
                "name": user["fullname"],
                "otp": otp
            }

            input_otp = {
                "email": email,
                "new_email": new_email,
                "otp": str(otp),
                "created_at": created_at
            }

            db.db.otp_change_email.insert_one(input_otp)

            msg = Message(subject="OTP to Verify New Email - Spinemotion", sender="spinemotionapp@gmail.com", recipients=[new_email])
            msg.html = render_template("verify-change-email.html", data=data)
            mail.send(msg)

            return {
                'message': "Berhasil request, cek email yang baru untuk lihat kode OTP"
            }, 200
        else:
            return { 'message': 'Password salah' }, 401
    except VerifyMismatchError:
        return { 'message': 'Password tidak sesuai' }, 401
```

Fungsi ini mengambil password dan email baru dari data JSON. Kemudian verifikasi password menggunakan PasswordHasher().verify(). Jika password valid, OTP acak dibuat dengan random.randint(100000, 999999) dan disimpan bersama data email baru, email pengguna dan timestamp di database dengan db.db.ootp_change_email.insert_one(input_otp). Setelah itu, email berisi OTP dikirim ke email baru menggunakan mail.send(msg). Fungsi mengembalikan respons 200 OK jika berhasil. atau 401 Unauthorized jika password tidak valid.

8. Verifikasi OTP

```python
@bp.route('/verify-otp', methods=['PUT'])
@jwt_required()
@api_key_required
def verify_new_email():
    data = request.get_json()
    return verify_otp_service(data)
```

Pengguna mengirimkan PUT request ke endpoint /user/verify-otp dengan data JSON berisi OTP yang mereka terima di email baru. Fungsi verify_otp_service(data) dipanggil untuk memverifikasi apakah OTP yang dimasukkan valid dan apakah alamat email baru dapat diperbarui.

```python
def verify_otp_service(data):
    otp = data["otp"]

    data_otp = db.db.otp_change_email.find_one({"otp": otp})

    if not data_otp:
        return {
            "message": "otp tidak sesuai"
        }, 400

    old_email = data_otp["email"]
    new_email = data_otp["new_email"]

    update_email = {
        "email": new_email
    }

    db.db.users.update_one({"email": old_email}, {"$set": update_email})
    db.db.otp_change_email.delete_one({"otp": otp})

    return {
        "message": "Email berhasil diubah"
    }, 200
```

OTP yang dikirimkan oleh pengguna diambil dari data[“otp”]. Fungsi mencari OTP yang diterima dalam koleksi otp_change_email di database. Jika OTP tidak ditemukan, mengembalikan status 400 (OTP tidak sesuai). Jika OTP valid, email lama dan email baru pengguna diambil dari data OTP yang ditemukan di database. Fungsi mempersiapkan data update_email yang berisi new_mail dan memperbarui email pengguna di koleksi users menggunakan db.db.users.update_one(). Setelah perubahan email berhasil, OTP yang telah digunakan dihapus dari koleksi otp_change_email untuk mencegah penyalahgunaan.Jika semuanya berhasil, fungsi mengembalikan status 200 OK (berhasil). Jika OTP tidak ditemukan, mengembalikan status 400 Bad Request.

9. Update Profil

```python
@bp.route('/update-profile', methods=['PUT'])
@jwt_required()
@api_key_required
def update_profile():
    data = request.form.to_dict()

    files=[]

    if "photo" in request.files:
        files = request.files["photo"]

    return update_profile_service(data, files)
```

@bp.route('/update-profile', methods=['PUT']) menangani permintaan PUT untuk memperbarui informasi profil pengguna. Pengguna mengirimkan permintaan untuk memperbarui informasi pribadi mereka seperti nama lengkap, nomor HP, dan foto profil. Kemudian data = request.form.to_dict() mengambil data yang dikirimkan oleh pengguna jika ada file foto yang diupload, data tersebut akan diambil dari request dan disimpan dalam variabel files.

```python
def update_profile_service(data, files):
    email = get_jwt_identity()
    user = db.db.users.find_one({"email": email})

    if not user:
        return {
            "message": "Pengguna tidak ditemukan"
        }, 404

    try:
        updates = {}
        if files:
            timestamp = datetime.now().strftime("%Y%m%d%H%M%S")
            filename = f"{timestamp}_image.jpg"
            path = f"static/uploads/profiles/{filename}"
            files.save(path)
            updates["photo"] = filename

        if "fullname" in data:
            updates["fullname"] = data["fullname"]

        if "no_hp" in data:
            updates["no_hp"] = data["no_hp"]

        if updates:
            db.db.users.update_one({"email": email}, {"$set": updates})

        return {
            "message": "Berhasil update profil"
        }
    except Exception as e:
        return {
            "message": f"Error {e}"
        }, 500
```

Fungsi email = get_jwt_identity() digunakan untuk mendapatkan email pengguna yang sedang login, yang diekstrak dari token JWT. user = db.db.users.find_one({"email": email}) mencari pengguna di database berdasarkan email yang diperoleh dari JWT. Jika pengguna tidak ditemukan, sistem akan mengembalikan status 404 (Not Found). Variabel updates = {} menyimpan data yang akan diperbarui di database (misalnya nama lengkap, nomor HP, dan foto profil). Jika ada file foto yang diupload, file tersebut akan disimpan dengan nama file yang diberi timestamp untuk memastikan nama file unik. Memeriksa nama lengkap (updates), no_hp (updates), perubahan data yang dikirimkan pengguna (update_one). Jika ada diupdate ke database. db.db.users.update_one({"email": email}, {"$set": updates}) memperbarui data pengguna di database sesuai dengan data yang ada di updates. Setelah pembaruan berhasil, fungsi mengembalikan pesan sukses. Jika terjadi kesalahan dalam proses akan mengembalikan respons status 500 (error).

10. Rekomendasi Gerakan

```python
@bp.route("/recomendation", methods=["POST"])
def get_rekomendasi():
    data = request.get_json()
    userId = data.get("userId")
    faktor_memperberat = data.get("faktor_memperberat", [])
    faktor_memperingan = data.get("faktor_memperingan", "")
    durasi = data.get("durasi", "")
    tingkat_nyeri = data.get("tingkat_nyeri", "")

    if not userId:
        return jsonify({"error": "userId harus disertakan"}), 400

    hasil = rekomendasi_gerakan(
        faktor_memperberat, faktor_memperingan, durasi, tingkat_nyeri
    )
    if hasil:
        diagnosa = {
            "faktor_memperberat": faktor_memperberat,
            "faktor_memperingan": faktor_memperingan,
            "durasi": durasi,
            "tingkat_nyeri": tingkat_nyeri,
        }
        simpan_rekomendasi(userId, hasil[0], diagnosa)
        return jsonify(hasil[0]), 200
    else:
        return jsonify({"message": "Tidak ada rekomendasi ditemukan"}), 404
```

Endpoint menggunakan method POST untuk menerima data gejala pengguna dalam format JSON. Fungsi get_rekomendasi() bertindak sebagai orchestrator. Dia akan mengekstrak data input (userId, faktor-faktor nyeri, dll) dari body request. Kemudian memmanggil fungsi rekomendasi_gerakan dari service layer untuk mendapatkan hasil rekomendasi. Jika rekomendasi ditemukan, ia akan memanggil fungsi simpan_rekomendasi untuk mencatat hasilnya ke database. Kemudian mengembalikan hasil rekomendasi pertama (hasil[0]) kepada klien dengan status 200 OK atau pesan 404 Not Found jika tidak ada rekomendasi yang cocok.

```python
@bp.route("/historyrecomendation", methods=["GET"])
def get_history():
    userId = request.args.get("userId")
    if not userId:
        return jsonify({"error": "Parameter userId dibutuhkan"}), 400

    data = get_history_by_userId(userId)
    return jsonify(data), 200
```

Endpoint menggunakan method GET untuk mengambil data riwayat. Ia menerima userId sebagai query parameter (/historyrecomendation?email=user@example.com). Fungsi ini memamnggil get_history_by_email dari service untuk melakukan kueri ke database dan mengembalikan hasilnya sebagai JSON.

```python
@bp.route("/historyrecomendation/<id>", methods=["DELETE"])
def delete_history(id):
    try:
        deleted = delete_history_by_id(id)
        if deleted:
            return jsonify({"message": "Riwayat berhasil dihapus"}), 200
        else:
            return jsonify({"error": "Riwayat tidak ditemukan"}), 404
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

Endpoint menggunakan method DELETE untuk operasi penghapusan. Dia menerima id riwayat yang akan dihapus sebagai path parameter di URL.

Kemudian memanggil delete_history_by_id dari service dan mengembalikan status 200 OK jika berhasil atau 404 Not Found jika ID riwayat tidak ada. Blok try...except menangani kemungkinan error lain.

```python
# Load dataset
current_dir = os.path.dirname(os.path.abspath(__file__))
file_path = os.path.abspath(
    os.path.join(current_dir, "..", "..", "model_weights", "dataset-rekomendasi.csv")
)
df = pd.read_csv(file_path, encoding="utf-8-sig")
df.columns = df.columns.str.strip().str.lower()

required_cols = [
    "faktor_memperberat",
    "faktor_memperingan",
    "durasi",
    "tingkat_nyeri",
    "arah_latihan",
    "gerakan1",
    "gerakan2",
    "gerakan3",
]
missing = [col for col in required_cols if col not in df.columns]
if missing:
    raise Exception(f"Missing columns: {missing}")

# Normalisasi kolom
for col in ["faktor_memperberat", "faktor_memperingan", "durasi", "tingkat_nyeri"]:
    df[col] = df[col].astype(str).str.strip().str.lower()

df["faktor_memperberat_list"] = df["faktor_memperberat"].apply(
    lambda x: [item.strip() for item in x.split(",")]
)
```

Bagian ini merupakan tahap inisialisasi dari service. Sistem menggunakan library Pandas untuk memuat dataset-rekomendasi.csv ke dalam sebuah DataFrame, yang berfungsi sebagai basis pengetahuan untuk sistem rekomendasi. Dilakukan beberapa langkah pra-pemrosesan data untuk memastikan konsistensi, seperti membersihkan nama kolom, mengubah semua data tekstual menjadi huruf kecil, dan mengubah string faktor_memperberat yang dipisahkan koma menjadi sebuah list/array untuk mempermudah proses pencocokan.

```python
def rekomendasi_gerakan(
    faktor_memperberat_user,
    faktor_memperingan_user,
    durasi_user,
    tingkat_nyeri_user,
    jumlah_rekomendasi=3,
):
    faktor_memperberat_user = [x.strip().lower() for x in faktor_memperberat_user]
    faktor_memperingan_user = faktor_memperingan_user.strip().lower()
    durasi_user = durasi_user.strip().lower()
    tingkat_nyeri_user = tingkat_nyeri_user.strip().lower()

    hasil = df.copy()
    hasil = hasil[hasil["tingkat_nyeri"] == tingkat_nyeri_user]
    hasil = hasil[hasil["durasi"] == durasi_user]
    hasil = hasil[hasil["faktor_memperingan"] == faktor_memperingan_user]
    hasil = hasil[
        hasil["faktor_memperberat_list"].apply(
            lambda x: any(item in x for item in faktor_memperberat_user)
        )
    ]

    rekomendasi = []
    for _, row in hasil.iterrows():
        rekomendasi.append(
            {
                "arah_latihan": row["arah_latihan"].capitalize(),
                "gerakan": [row["gerakan1"], row["gerakan2"], row["gerakan3"]],
            }
        )
    return rekomendasi[:jumlah_rekomendasi]
```

Fungsi rekomendasi_gerakan menerapkan logika rule-based system dengan melakukan filtering DataFrame secara bertahap.

DataFrame difilter berdasarkan kecocokan persis pada tingkat_nyeri, durasi, dan faktor_memperingan. Untuk faktor_memperberat, logikanya lebih fleksibel, yaitu mencari baris dimana setidaknya salah satu faktor dari pengguna cocok dengan daftar faktor pada dataset. Hasil yang telah difilter kemudian diformast dan dikembalikan sebagai daftar rekomendasi.

```python
def simpan_rekomendasi(userId, rekomendasi, diagnosa):
    timestamp = datetime.now(ZoneInfo("Asia/Jakarta")).isoformat()
    db.db.recomendation.insert_one(
        {
            "userId": userId,
            "timestamp": timestamp,
            "rekomendasi": rekomendasi,
            "diagnosa": diagnosa,
        }
    )

def get_history_by_userId(userId):
    results = db.db.recomendation.find({"userId": userId})
    data = []
    for doc in results:
        doc["_id"] = str(doc["_id"])
        data.append(doc)
    return data

def delete_history_by_id(history_id):
    from bson import ObjectId
    result = db.db.recomendation.delete_one({"_id": ObjectId(history_id)})
    return result.deleted_count > 0
```

simpan_rekomendasi akan melakukan insert_one ke koleksi recomendation untuk menyimpan jejak setiap rekomendasi yang berhasil diberikan kepada pengguna.
get_history_by_userId akan melakukan find untuk mengambil semua dokumen dari koleksi recomendation yang cocok dengan userId tertentu.
deleted_history_by_id akan menerima history_id dalam format string, mengubahnya menjadi delete_one. Fungsi ini mengembalikan nilai True atau False berdasarkan result.deleted_count untuk mengonfirmasi apakah ada dokumen yang benar-benar terhapus.

11. Deteksi Gerakan

```python
class PoseLSTM(nn.Module):
    def __init__(self, input_size=132, hidden_size=128, num_classes=12):
        super(PoseLSTM, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        _, (h_n, _) = self.lstm(x)
        return self.fc(h_n[-1])

DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
MODEL_PATH = "./model_weights/final_lstm_pose_model.pt"
ENCODER_PATH = "./model_weights/final_label_encoder.pkl"

with open(ENCODER_PATH, "rb") as f:
    label_encoder = pickle.load(f)
    print(f"[DEBUG] Label encoder berisi kelas: {label_encoder.classes_}")

num_classes = len(label_encoder.classes_)
model = PoseLSTM(num_classes=num_classes).to(DEVICE)
model.load_state_dict(torch.load(MODEL_PATH, map_location=DEVICE))
model.eval()
print(f"[INFO] Model berhasil dimuat. Perangkat: {DEVICE}")

print("[INFO] Inisialisasi MediaPipe dan buffer pengguna...")

mp_holistic = mp.solutions.holistic
mp_drawing = mp.solutions.drawing_utils
user_buffers = defaultdict(lambda: deque(maxlen=30))
```

PoseLST mendefinisikan arsitektur neural network menggunakan PyTorch, yang terdiri dari lapisan LSTM dan sebuah fully-connected layer (Linear) untuk klasifikasi. Model yang dilatih sebelumnya (.pt) dimuat kedalam memori. pickle-load(f) memuat label_encoder (menerjemahkan output numerik dari model menjadi nama kelas gerakan yang dapat dibaca). mp_holistic diinisialisasi. Komponen ini akan digunakan untuk mengekstrak 33 titik kunci tubuh (landmarks) dari setiap frame video. user_buffers menyimpan buffer (antrean dengan panjang maksimal 30 frmae) untuk setiap pengguna, karena model LSTM memerlukan sekuens data untuk melakukan prediksi. user_predictions adalah buffer kedua yang lebih kecil (5 prediksi terakhir) yang digunakan untuk menstabilkan hasil prediksi nanti.

```
socketio.on_event("connect", handle_connect)
socketio.on_event("disconnect", handle_disconnected)
socketio.on_event("message", handle_message)
socketio.on_event("image", handle_image)
```

Sistem menggunakan protokol WebSocket via Flask-SocketIO untuk komunikasi real-time. Kode ini mendaftarkan fungsi handler untuk setiap event yang diterima dari klien. Event yang paling krusial adalah image, yang akan memicu fungsi handler)image setiap kali mengirimkan sebuah frame video.

```
def handle_image(data):
    try:
        print("[INFO] Menerima data gambar dari klien...")

        user_id = data.get("userId")
        selected_pose = data.get("selected_pose")
        image_data = data.get("image_data")

        if not all([user_id, selected_pose, image_data]):
            print("[ERROR] Data tidak lengkap:", data)
            raise ValueError("Data tidak lengkap dari klien")

        print(f"[DEBUG] User ID: {user_id}, Pose yang dipilih: {selected_pose}")

        image_bytes = base64.b64decode(image_data)
        np_image = np.frombuffer(image_bytes, np.uint8)
        image = cv2.imdecode(np_image, cv2.IMREAD_COLOR)
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
            results = holistic.process(image_rgb)
            image_bgr = cv2.cvtColor(image_rgb, cv2.COLOR_RGB2BGR)

            landmarks = results.pose_landmarks.landmark
            pose_row = np.array([[lmk.x, lmk.y, lmk.z, lmk.visibility] for lmk in landmarks]).flatten()
            pose_row = np.clip(pose_row, 0.0, 1.0)

            buffer = user_buffers[user_id]
            buffer.append(pose_row)

            print(f"[DEBUG] Buffer panjang saat ini: {len(buffer)}")

            if len(buffer) < 30:
                emit("response", {
                    "status": "buffering",
                    "progress": len(buffer)
                })
                return

            input_seq = np.array(buffer)[np.newaxis, ...]
            input_tensor = torch.from_numpy(input_seq).float().to(DEVICE)

            input_seq = np.array(buffer)[np.newaxis, ...]
            input_tensor = torch.from_numpy(input_seq).float().to(DEVICE)

            status = "Sesuai" if pose_class == selected_pose else "Tidak Sesuai"

            db.db.detections.insert_one({
                "userID": user_id,
                "namaGerakan": selected_pose,
                "keterangan": status,
                "gender": gender,
                "tanggal": datetime.today().strftime("%Y-%m-%d %H:%M:%S"),
            })

            print(f"[INFO] Data disimpan ke DB: Pose: {selected_pose}, Status: {status}")

            _, buffer_img = cv2.imencode(".jpg", image_bgr)
            encoded_image = base64.b64encode(buffer_img).decode("utf-8")

            emit("response", {
                "status": "ok",
                "pose_class": pose_class,
                "prob": str(float(np.max(pose_probabilities))),
                "imageData": encoded_image,
            })

            print("[INFO] Respon dikirim ke klien.")

    except Exception as e:
        print("[ERROR] Terjadi kesalahan di handle_image:", e)
        emit("response", {"status": "error", "message": str(e)})
```

handler_image menerima data gambar (Base64) dari klien dan mengubahnya menjadi format gambar yang dapat diolah oleh OpenCV. Gambar diumpankan ke Mediapipe Holistic yang kemudian mengembalikan posisi 33 titik tubuh dalam koordinat (x, y, z). Ini adalah proses feature extraction. Koordinat tersebut "diratakan" menjadi sebuah array tunggal dan dimasukkan ke dalam user_buffers. Proses ini diulang terus-menerus. Jika buffer belum berisi 30 frame, server hanya mengirim status "buffering" ke klien. Setelah buffer terisi 30 frame, sekuens data ini dimasukkan ke dalam model LSTM.

Model menganalisis pergerakan dari waktu ke waktu dalam 30 frame tersebut untuk menghasilkan satu prediksi pose. Untuk mencegah hasil prediksi "bergetar" atau flickering (berganti-ganti dengan cepat antara dua pose) sistem menyimpan 5 prediksi terakhir. Prediksi yang paling sering muncul dalam 5 riwayat terakhir itulan yang dianggap sebagai hasil akhir (stable_prediction). Prediksi stabil ini dibandingkan dengan pose yang seharusnya dilakukan pengguna untuk menentukan status :Sesuai? atau "Tidak Sesuai". Hasil deteksi ini dismpan di database MongoDB, dan respons lengkap (termasuk status, nama pose, dan gambar dengan landmark yang digambar ulang) diirim kembali ke klien untuk memberikan umpan balik visual dan tekstual secara real-time.

12. Feedback Latihan

```python
@bp.route("/feedback", methods=["POST"])
def post_feedback():
    data = request.get_json(silent=True)
    if not data:
        return jsonify({"error": "Invalid JSON"}), 400

    userId = data.get("userId")
    date = data.get("date")
    daftar_gerakan = data.get("daftar_gerakan", [])
    pain = data.get("pain_level")

    if not userId or not date or pain is None:
        return jsonify({"error": "Fields email, date, pain_level dibutuhkan"}), 400

    try:
        pain = int(pain)
        if not (0 <= pain <= 10):
            raise ValueError()
    except (ValueError, TypeError):
        return jsonify({"error": "pain_level harus integer 0-10"}), 400

    result = simpan_feedback(userId, date, daftar_gerakan, pain)
    return (
        jsonify(
            {"message": "Feedback received", "feedback_id": str(result.inserted_id)}
        ),
        201,
    )
```

Endpoint menggunakan POST untuk menerima data dalam format JSON dari body request. Sebelum memproses, sistem melakukan beberapa validasi penting, diantaranya memastikan semua field (userId, date, pain_level) ada didalam request, memvalidasi tipe data dan rentang nilai untuk pain_level, yang harus berupa integer antara 0 dan 10. Jika data valid, fungsi ini memanggil simpan_feedback dari service layer untuk melakukan operasi penyimpanan ke database. Setelah berhasil disimpan, sistem mengembalikan respons 201 Created yang berisi pesan sukses dan ID unik dari dokumen feedback yang berisi pesan sukses dan ID unik dari dokumen feedback yang baru saja dibuat.

```python
@bp.route("/feedback", methods=["GET"])
def get_feedback():
    userId = request.args.get("userId")
    if not userId:
        return jsonify({"error": "Parameter userId diperlukan"}), 400

    feedbacks = ambil_feedback_by_userId(userId)
    return jsonify(feedbacks), 200
```

Endpoint menggunakan method GET dan menerima userId sebagai query parameter dari URL (/feedback?userId=12345). Fungsi ini memanggil ambil_feedback_by_userId untuk mengambil data dari database. Dia akan mengembalikan sebuah array berisi semua dokumen umpan balik yang ditemukan sebagai JSON dengan status 200 OK.

```python
def simpan_feedback(userId: str, date: str, daftarGerakan: str, pain_level: int):
    """
    Simpan satu dokumen feedback pain scale ke koleksi `feedback`.
    """
    doc = {
        "userId": userId,
        "date": date,
        "daftar_gerakan": daftarGerakan,
        "pain_level": pain_level,
    }
    result = db.db.feedback.insert_one(doc)
    return result
```

Fungsi simpan_feedback menyusun sebuah dictionary Python (yang akan menjadi dokumen BSON di MongoDB) dari argumen yang diterima. Kemudian melakukan operasi insert_one ke koleksi feedback di database MongoDB. Kemudian objek result dari PyMongo, yang salah satunya berisi inserted_id dari dokumen baru.

```python
def ambil_feedback_by_userId(userId: str):
    cursor = db.db.feedback.find({"userId": userId})
    feedbacks = []
    for doc in cursor:
        feedbacks.append(
            {
                "id": str(doc["_id"]),
                "userId": doc["userId"],
                "date": doc["date"],
                "painLevel": doc["pain_level"],
                "daftar_gerakan": doc.get("daftar_gerakan", []),  # <- tambahkan ini
            }
        )
    return feedbacks
```

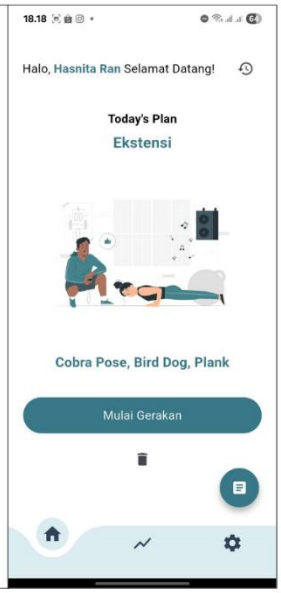
Fungsi melakukan operasi baca (find) untuk mengambil semua dokumen dari koleksi feedback yang cocok dengan userId. Sistem melakukan iterasi pada setiap dokumen yang ditemukan dan melakukan beberapa hal yaitu, mengubah tipe data _id dari ObjectId MongoDB menjadi string agar kompatibel dengan format JSON, mengubah nama field (pain_level menjadi painLevel) untuk konsistensi di sisi klien, menggunakan doc.get("daftar_gerakan", []) untuk mencegah error jika ada dokumen lama yang tidak memiliki field tersebut. Kemudian fungsi mengembalikan daftar (array) berisi objek-objek feedback yang sudah diformat dengan rapi.

IMPLEMENTASI MOBILE

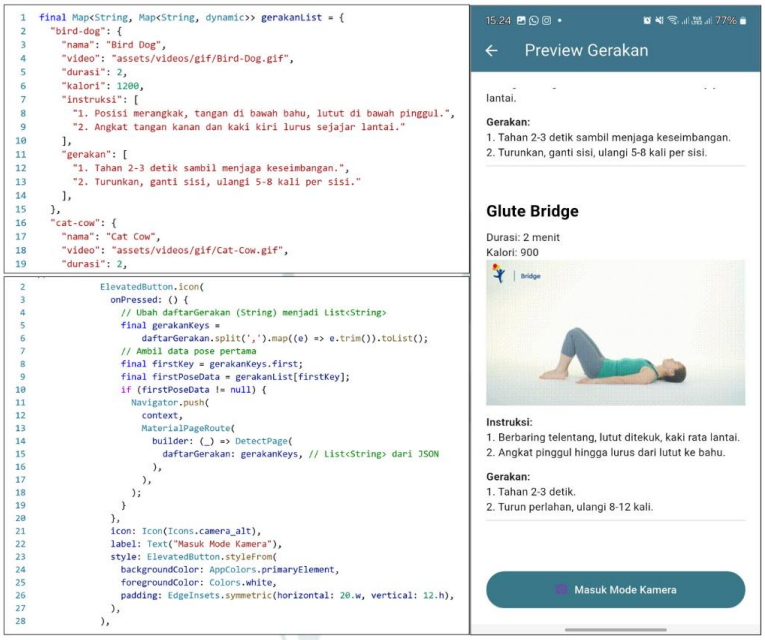
```dart
// Tombol Selesai
                Expanded(
                  child: ElevatedButton(
                    onPressed: provider.data.yangMeringankan != null
                        ? () async {
                            final email = Provider.of<LoginProvider>(context,
                                    listen: false)
                                .userEmail!;
                            await provider.submitDiagnosis(email);
                            final rekomendasi =
                                provider.resultRekomendasi ?? "-";
                            final gerakan = provider.resultGerakan ?? "-";
                            Provider.of<RecomendationProvider>(context,
                                    listen: false)
                                .setFromText(rekomendasi, gerakan);
                            Navigator.pushNamedAndRemoveUntil(
                              context,
                              '/home',
                              (Route<dynamic> route) => false,
                            );
                          }
                        : null,
                    style: ElevatedButton.styleFrom(
                      backgroundColor: AppColors.primaryElement,
                      foregroundColor: Colors.white,
                      padding: const EdgeInsets.symmetric(vertical: 10),
                      shape: RoundedRectangleBorder(
                        borderRadius: BorderRadius.circular(12),
                      ),
                    ),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.center,
                      children: const [
                        Text("Selesai"),
                        SizedBox(width: 13),
                        Icon(Icons.check, color: Colors.white),
                      ],
                    ),
                  ),
                ),
```

Logika dieksekusi didalam onPressed pada tombol "Selesai" di halaman terakhir kuesioner. Kode ini mengorkestrasi beberapa Provider untuk mengumpulkan semua data yang diperlukan, diantaranya dia akan mengambil jawaban-jawaban kuesioner yang sudah tersimpan di dalam DiagnosisProvider, ia mengambil userId dari pengguna yang sedang login melalui DatabaseProvider. DatabaseProvider sebelumnya telah mengambil ID ini dari penyimpanan lokal (SharedPreferences). Semua data yang terkumpul diteruskan sebagai parameter saat memanggil fungsi fetchRecomendation, yang bertanggung jawab untuk berkomunikasi dengan backend.

```dart
Future<void> fetchRecomendation({
    required List<String> faktorMemperberat,
    required String faktorMemperingan,
    required String durasi,
    required String tingkatNyeri,
  }) async {
    _isLoading = true;
    notifyListeners();

    final url = "$_url/recomendation";

    try {
      final response = await http.post(
        Uri.parse(url),
        headers: {"Content-Type": "application/json"},
        body: jsonEncode({
          "faktor_memperberat": faktorMemperberat,
          "faktor_memperingan": faktorMemperingan,
          "durasi": durasi,
          "tingkat_nyeri": tingkatNyeri,
        }),
      );

      if (response.statusCode == 200) {
        final Map<String, dynamic> rekom = jsonDecode(response.body);

        _recomendation = RecomendationModel(
          rekomendasi: rekom['arah_latihan'] ?? '',
          gerakan: (rekom['gerakan'] as List<dynamic>).join(", "),
        );

        _resMessage = "Berhasil dapat rekomendasi";
      } else {
        _resMessage = "Gagal mendapatkan rekomendasi.";
        _recomendation = null;
      }
    } catch (e) {
      _resMessage = "Terjadi kesalahan: $e";
      _recomendation = null;
    }

    _isLoading = false;
    notifyListeners();
  }
```



Fungsi fetchRecomendation mengatur _isLoading menjadi true untuk menampilkan loading di UI. Dengan menggunakan linrary http untuk membuat permintaan POST ke endpoint backend /recomendation. Semua data dari kuesioner dan userId di-encode menjadi sebuah string JSON dan dikirim sebagai body dari permintaan. Struktur JSON ini dibuat agar sesuai persis dengan yang diharapkan oleh backend API. Setelah respons diterima, kode memeriksa statusCode. Jika 200 (sukses), body JSON dari respons di-decode dan hasilnya disimpan ke dalam state _recomendation. State _isLoading kemudian dikembalikan ke false untuk menyembunyikan indikator loading.



```
final Map<String, Map<String, dynamic>> gerakanList = {
  "bird-dog": {
    "nama": "Bird Dog",
    "video": "assets/videos/gif/Bird-Dog.gif",
    "durasi": 2,
    "kalori": 1200,
    "instruksi": [
      "1. Posisi merangkak, tangan di bawah bahu, lutut di bawah pinggul.",
      "2. Angkat tangan kanan dan kaki kiri lurus sejajar lantai."
    ],
    "gerakan": [
      "1. Tahan 2-3 detik sambil menjaga keseimbangan.",
      "2. Turunkan, ganti sisi, ulangi 5-8 kali per sisi."
    ],
  },
  "cat-cow": {
    "nama": "Cat Cow",
    "video": "assets/videos/gif/Cat-Cow.gif",
    "durasi": 2,
```

```
          ElevatedButton.icon(
            onPressed: () {
              // Ubah daftarGerakan (String) menjadi List<String>
              final gerakanKeys =
                  daftarGerakan.split(',').map((e) => e.trim()).toList();
              // Ambil data pose pertama
              final firstKey = gerakanKeys.first;
              final firstPoseData = gerakanList[firstKey];
              if (firstPoseData != null) {
                Navigator.push(
                  context,
                  MaterialPageRoute(
                    builder: (_) => DetectPage(
                      daftarGerakan: gerakanKeys, // List<String> dari JSON
                    ),
                  ),
                );
              }
            },
            icon: Icon(Icons.camera_alt),
            label: Text("Masuk Mode Kamera"),
            style: ElevatedButton.styleFrom(
              backgroundColor: AppColors.primaryElement,
              foregroundColor: Colors.white,
              padding: EdgeInsets.symmetric(horizontal: 20.w, vertical: 12.h),
            ),
          ),
```


Arsitektur ini memisahkan logika rekomendasi (di backend) dengan aset konten (di frontend). Backend hanya bertugas memberikan kunci, contohnya {"gerakan": "cat-cow", :bird-dog", "plank"]}. Aplikasi menerima daftar kunci tersebut. Untuk setiap kunci (misal: "cat-cow") , aplikasi akan mencarinya didalam Map lokal gerakanList. Setelah data detail (nama, path video/gif, instruksi) ditemukan di gerakanList, barulah aplikasi me-render antarmuka pengguna yang kaya, seperti menampilkan gambar gif, durasi, dan langkah-langkah instruksi dalam sebuah ListView atau Card.

Saat tombol ditekan pada (ElevatedButton), aplikasi melakukan navigasi ke DetectPage. Yang terpenting adalah daftarGerakan (sebuah List berisi kunci-kunci gerakan seperti ["cat-cow", "bird-dog"]) diteruskan sebagai argumen ke DetectPage. Halaman deteksi kini tahu gerakan yang harus dipandu

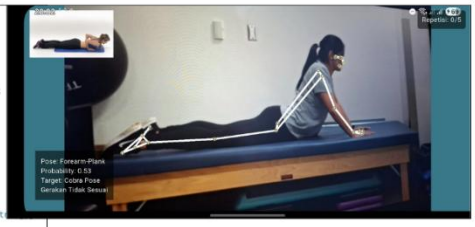
```dart
Future<void> _initCameraAndSocket() async {
  // Kamera
  try {
    final cams = await availableCameras();
    final front = cams.firstWhere(
        (c) => c.lensDirection == CameraLensDirection.front,
        orElse: () => cams.first);
    _controller = CameraController(front, ResolutionPreset.high);
    await _controller!.initialize();
    setState(() => _isCameraReady = true);
  } catch (e) {
    debugPrint('Camera init error: $e');
    return;
  }

  // Socket & userId
  userId = await DatabaseProvider().getUserId();
  _socket = IO.io(ApiEndPoints.baseUrl, {
    'transports': ['websocket'],
    'autoConnect': false,
  });
  _socket.on('connect', (_) {
    setState(() => _isConnected = true);
    _startFrameLoop();
    if (USE_TIMERS) _startCountdown();
    ScaffoldMessenger.of(context)
        .showSnackBar(const SnackBar(content: Text('Terhubung ke server')));
  });
  _socket.on('response', _handleServerResponse);
  _socket.on('disconnect', (_) => setState(() => _isConnected = false));
  _socket.connect();
}
```

Menggunakan library camera, aplikasi meminta akses ke kamera perangkat, memilih kamera belakang, dan mengaturnya pada resolusi tinggi. Menggunakan library socket_io_client, aplikasi membuat instance koneksi WebSocket yang diarahkan ke url server backend. Sebelum koneksi dimulai aplikasi mendaftarkan handler untuk tiga event utama: connect (untuk memulai pengiriman frame saat terhubung), disconnect, dan yang paling penting, response yang akan memanggil fungsi _handlerServerResponse setiap kali server mengirimkan data analisis.

```dart
void _startFrameLoop() {
  _frameTimer?.cancel();
  _frameTimer = Timer.periodic(const Duration(seconds: 2), (_) async {
    if (_controller == null || !_controller!.value.isInitialized) return;
    final pic = await _controller!.takePicture();
    final bytes = await File(pic.path).readAsBytes();
    _socket.emit('image', {
      'image_data': base64Encode(bytes),
      'userId': userId,
      'selected_pose': _selectedPose,
    });
  });
}
```

Timer diatur untuk berjalan setiap 2 detik. Ini mengontrol frekuensi pengiriman data agar tidak membebani server dan jaringan. Di setiap interval, aplikasi menangkap satu frame gambar dari kamera, membacanya sebagai bytes, dan meng-encode nya ke dalam format Base64. Data gambar Base64 tersebut, beserta userId dan nama gerakan target (_selectedPose), dikirimkan ke server melalui Socket.IO event yang bernama ‘image’. Data ini akan diterima dan diproses oleh handler @socketio.on(‘image’) di sisi backend.

```dart
void _handleServerResponse(dynamic data) {
    try {
      final status = data['status'] ?? 'ok';

      if (status == 'buffering') {
        final progress = data['progress'] ?? 0;
        debugPrint('⏳ Buffering frame: $progress/30');
        return; // jangan proses apa-apa dulu
      }

      if (status == 'no_pose') {
        debugPrint('🚫 Tidak ada pose terdeteksi.');
        return;
      }

      // Status OK: proses hasil prediksi
      final bytes = base64Decode(data['imageData'] as St
      final bodyClass = data['pose_class'] as String;
      final prob = double.parse(data['prob'].toString());

      if (!mounted) return;
      setState(() {
        _imageBytes = Uint8List.fromList(bytes);
        _poseClass = bodyClass;
        _probability = prob;
        _showImage = true;
      });

      if (isMatchingPose(bodyClass, _selectedPose) &&
          _repetitions < _maxRepetitions &&
          prob > 0.8) {
        _repetitions++;
        if (_repetitions >= _maxRepetitions) _nextPose();
      }
    } catch (e) {
      debugPrint('Response handling error: $e');
    }

    _hideImageTimer?.cancel();
    _hideImageTimer = Timer(const Duration(seconds: 1), () {
      if (!mounted) return;
      setState(() => _showImage = false);
    });
  }
```



Fungsi ini menerima data JSON dari server, lalu mem-parsing informasi penting seperti gambar yang sudah dianotasi (imageData), nama pose yang terdeteksi (pose_class), dan tingkat kepercayaan (prob). setState dipanggil untuk memperbarui antarmuka pengguna secara real-time, menampilkan gambar dengan landmark dari server dan hasil prediksinya. Sistem secara aktif memeriksa apakah pose yang terdeteksi (bodyClass) sesuai dengan pose yang seharusnya dilakukan (_selectedPose). Jika sesuai dan model cukup yakin (prob > 0.7), maka hitungan repetisi (_repetitions) akan bertambah. Ini mengubah fitur dari sekadar "deteksi" menjadi "sistem pemandu latihan" yang interaktif. Jika target repetisi tercapai, fungsi _nextPose akan dipanggil.

```dart
Future<void> _sendPainFeedback(
      String userId, int level, List<String> daftarGerakan)
    final now = DateTime.now().toIso8601String();
    try {
      final res = await http.post(
        Uri.parse('${ApiEndPoints.baseUrl}/feedback'),
        headers: {'Content-Type': 'application/json'},
        body: jsonEncode({
          'userId': userId,
          'date': now,
          'daftar_gerakan': daftarGerakan, // <-- tambahkan
          'pain_level': level,
        }),
      );

      debugPrint(res.statusCode == 200
          ? 'Feedback berhasil'
          : 'Feedback gagal: ${res.statusCode}');
    } catch (e) {
      debugPrint('Error kirim feedback: $e');
    }
  }
```

Setelah pengguna memasukkan tingkat nyeri mereka melalui AlertDialog, fungsi _sendPainFeedback dieksekusi. Fungsi ini membuat POST request ke endpoint backend /feedback. Payload JSON berisi userId, timedtamp saat ini, daftar gerakan yang baru saja dilakukan, dan tingkat nyeri yang diinput pengguna. Data ini kemudian disimpan di database untuk analisis.

```dart
Future<void> fetchFeedback() async {
    if (_daftarGerakan.isEmpty) {
      if (mounted) setState(() => _isLoading = false);
      return;
    }
    ;

    final uri =
        Uri.parse("${ApiEndPoints.baseUrl}/feedback?email=${widget.email}");
    final response = await http.get(uri);
    final inputPoseSet = List<String>.from(_daftarGerakan)..sort();

    if (response.statusCode == 200) {
      final List data = json.decode(response.body);
      List<FeedbackModel> list =
          data.map((e) => FeedbackModel.fromJson(e)).toList();

      final filtered = list.where((feedback) {
        final itemSet = List<String>.from(feedback.daftarGerakan)..sort();
        return const ListEquality().equals(inputPoseSet, itemSet);
      }).toList();

      final filteredByRange = _filterByRange(_selectedRange, filtered);

      if (mounted) {
        setState(() {
          _feedbackList = filtered;
          _filteredList = filteredByRange;
          _isLoading = false;
        });
      }
    } else {
      if (mounted) {
        setState(() {
          _isLoading = false;
        });
      }
    }
  }
```

fetchFeedback melakukan GET request ke endpoin /feedback untuk menarik seluruh riwayat umpan balik pengguna. Sistem hanya akan menganalisis data yang berasal dari sesi latihan dengan set gerakan yang sama persis. Ini dilakukan dengan membandingkan List gerakan dari setiap entri riwayat dengan set gerakan yang sedang aktif. Pengguna ListEquality().equals memastikan perbandingan "apel dengan apel", sehingga analisis tren menjadi valid. Data yang sudah relevan kemudian disaring lagi berdasarkan rentang waktu yang dipilih pengguna (misal: "1 Minggu Terakhir"), yang diimplementasikan dalam fungsi _filterByRange. Hasil akhir dari proses ini adalah _filteredList, data yang siap untuk dianalisis dan divisualisasikan.

```dart
    // Aturan 1: Nyeri Melebihi Batas Aman (misal > 7)
    for (var feedback in feedbackList) {
      final painLevel = double.tryParse(feedback.painLevel.toString()) ?? 0.0;
      if (painLevel > 7) {
        return _buildResult(
          feedbackList,
          'Nyeri sangat tinggi! Segera konsultasi ke dokter.',
          Colors.red.shade800,
        );
      }
    }
```

```dart
    // Aturan 3: Tren Nyeri yang Konsisten Memburuk
    final trend = _calculateTrend(feedbackList);
    final int naik = trend['naik']!;
    final int turun = trend['turun']!;
    if (naik > turun && feedbackList.length > 3) {
      return _buildResult(
        feedbackList,
        'Nyeri cenderung meningkat. Konsultasi ke dokter.',
        Colors.orange.shade800,
      );
    }
```

```dart
    return _buildResult(
      feedbackList,
      'Progres yang baik! Lanjutkan latihan ini.',
      const Color(0xFF47828D),
    );
```

Fungsi analyzePainData menerapkan serangkaian aturan secara berurutan pada data tingkat nyeri.

Ia memeriksa kondisi-kondisi berbahaya terlebih dahulu (seperti tingkat nyeri absolut yang sangat tinggi atau tren yang terus memburuk). Jika tidak ada kondisi berbahaya yang terdeteksi, sistem akan memberikan rekomendasi positif. Hasil analisis (teks rekomendasi, warna teks, dan data statistik lainnya) dibungkus dalam sebuah objek AnalysisResult.

```
                child: RepaintBoundary(
                  key: _chartKey,
                  child: Container(
                    color: themeProvider.backgroundColor,
                    padding: const EdgeInsets.only(
                        right: 16, top: 16),
                    child: LineChart(
                      LineChartData(
                        minY: 1,
                        maxY: 10,
                        minX: -0.5,
                        maxX: _filteredList.length.toDouble() - 0.5,
                        gridData: FlGridData(
                          show: true,
                          drawVerticalLine: true,
                          drawHorizontalLine: true,
                          horizontalInterval: 2,
                          verticalInterval: 1,
                          getDrawingHorizontalLine: (value) {
                            return FlLine(
                              color: Colors.grey.withOpacity(0.3),
                              strokeWidth: 1,
                        lineBarsData: [
                          LineChartBarData(
                            spots: _filteredList.asMap().entries.map((entry)
                              double y = double.tryParse(entry.value.painLe
                              return FlSpot(entry.key.toDouble(), y);
                            }).toList(),
                            isCurved: true,
                            color: const Color(0xFF47828D),
```

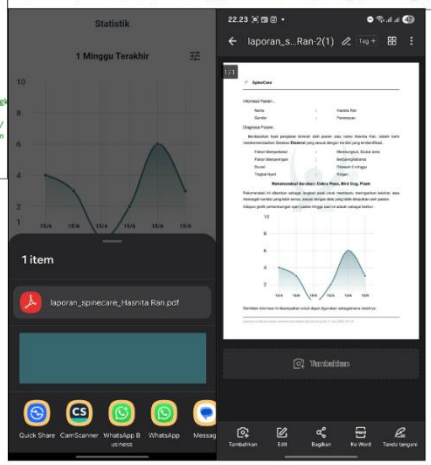
fl_chart, data _filteresList di map menjadi FlSpot (koordinat x,y) untuk di-plot sebagai grafik garis. Ini memberikan representasi visual yang intuitif dari tren penurunan atau kenaikan nyeri pengguna dari waktu ke waktu.

```
Future<void> generatePdfReport() async {
    final chartBytes = await _captureChart();

    final pdf = pw.Document();
    final profileProvider =
        Provider.of<ProfileProvider>(context, listen: false);
    final fullname =
```

```
        pw.Center(
          child: pw.Image(
            pw.MemoryImage(chartBytes),
            width: 350, // atau atur ke ukuran idealmu
            fit: pw.BoxFit.contain,
          ),
        ),
```

```
pdf.addPage(pw.Page(
  pageFormat: PdfPageFormat.a4,
  margin: const pw.EdgeInsets.symmetric(
    horizontal: 60,
    vertical: 50,
  ),
  build: (ctx) => pw.Stack(children: [
    // Watermark sebagai background
    pw.Positioned.fill(
      child: pw.Center(
        child: pw.Opacity(
          opacity: 0.1, // Sesuaikan ting
          child: pw.Image(
            pw.MemoryImage(logoBytes), //
            width: 600, // atau sesuaikan
            fit: pw.BoxFit.scaleDown,
          ),
        ),
```

```
  await Printing.sharePdf(
      bytes: await pdf.save(), filename: 'laporan_spinecare_${fullname}.pdf');
```



Pengambilan gambar grafik menggunakan RepaintBoundary dan _chartKey, aplikasi "mengambil screenshot" dari widget grafik yang sedang ditampilkan di layar. Fungsi generatedPdfReport mengumpulkan semua informasi relevan seperti data profil pengguna, detail diagnosis awal, dan gambar grafik yang baru saja diambil. Pada proses render pdf dia menggunakan library pdf dan printing, semua informasi ini disusun dalam format laporan A4. Hasilnya kemudian dapat dibagikan atau disimpan oleh pengguna sebagai file pdf, berfungsi sebagai laporan medis portabel.

Lampiran 7 Sertifikat HKI

REPUBLIK INDONESIA
KEMENTERIAN HUKUM

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC002025077793, 28 Juni 2025
Pencipta		
Nama	:	**Hasnita Rani Kumala, Dwi Intan Af'idah, S.T., M.Kom. dkk**
Alamat	:	Jl. Sunan Gunung Jati 1. RT/RW 001/002, Kelurahan Limbangan Wetan, Brebes, Kab. Brebes, Jawa Tengah, 52218
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Pusat Penelitian dan Pengabdian Masyarakat (P3M) Politeknik Harapan Bersama**
Alamat	:	Jalan Mataram No. 9, Pesurungan Lor, Kecamatan Margadana, Margadana, Kota Tegal, Jawa Tengah, 52142
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Aplikasi Panduan Gerakan Terapi untuk Mengurangi Nyeri Punggung Bawah Berbasis Mobile**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	28 Juni 2025, di Kota Tegal
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor Pencatatan	:	000918054

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL
u.b
Direktur Hak Cipta dan Desain Industri

Agung Damarsasongko,SH.,MH.
NIP. 196912261994031001

Balai Besar Sertifikasi Elektronik

Disclaimer:
1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah disegel secara elektronik menggunakan segel elektronik yang ditertibkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Hasnita Rani Kumala	Jl. Sunan Gunung Jati 1. RT/RW 001/002, Kelurahan Limbangan Wetan Brebes, Kab. Brebes
2	Dwi Intan Af'idah, S.T., M.Kom.	Desa Grinting RT/RW 003/001 Bulakamba, Kab. Brebes
3	Hepatika Zidny Ilmadina, S.Pd., M.Kom.	Jalan Kenanga Gang I Nomor 9, Kelurahan Mangkukusuman Tegal Timur, Kota Tegal

Lampiran 8 Lembar Bimbingan Skripsi

D IV TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN TUGAS AKHIR

Nama : Hasnita Rani Kumala
NIM : 21090092
No.Ponsel : 081229904855
Judul TA : Aplikasi Panduan Gerakan Fisioterapi Untuk Mencegah dan Mengurangi Low Back Pain Berbasis Mobile

Dosen Pembimbing I : Dwi Intan Af'idah, S.T., M.Kom

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1	21/3 - 25	konsep aplikasi	Identifikasi kebutuhan dengan tepat	
2.	17/4 - 25	model	mencoba pemodelan dg metode terkini	
3.	25/5 - 25	model dan aplikasi	penerapan model dan fitur history	
4.	2/6 - 25	model dan aplikasi	Dicoba untuk pemodelan yang lebih akurat	
5.	13/6 - 2025	Aplikasi	- Perhitungan cendarung naik dan cendarung turun pada grafik	

6.	20/6-2025	Aplikasi	- Sudah cukup - lanjut ke hak cipta - laporan	
7.	29/6-2025	laporan	perbaiki sesuai catatan	
8.	8/7-2025	laporan	perbaiki sesuai catatan	
9.	10/7-2025	laporan	ACC	

Tegal, 15 Juli 2025
Dosen Pembimbing I

Dwi Intan Af'idah, S.T., M.Kom.
NIPY. 11.020.470

D IV TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN TUGAS AKHIR

Nama : Hasnita Rani Kumala
NIM : 21090092
No.Ponsel : 081229904855
Judul TA : Aplikasi Panduan Gerakan Fisioterapi Untuk Mencegah dan Mengurangi Low Back Pain Berbasis Mobile

Dosen Pembimbing II : Hepatika Zidny Ilmadina, S.Pd., M.Kom.

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1	21/2025/03	* Observasi	* perlu pendalaman observasi kasus LBP. Terutama terkait skala nyeri atau derajat sakit, urutan gerakan, dan faktor berpengaruh lainnya sebagai acuan perancangan aplikasi.	
2.	23/2025/04	* Observasi Dataset	* penyusunan faktor-faktor keamanan pengguna sebelum menggunakan aplikasi. serta kombinasi dataset penentuan gerakan berdasarkan kondisi pengguna urutan atribut yang paling berpengaruh dan ditampilkan secara berlahap.	

3.	15/05/2025	* aplikasi	* flow udah sesuai dg keadaan user * belum demo aplikasi (1 flow aplikasi y/ 1 user)	
4.	26/05/2025	* aplikasi	* dari gerakan akhir itu ada halaman "closing" * history tidak boleh y/ mengulang gerakan lagi ; kecuali diberikan history hasil kuesionernya saja	
5.	12/06/2025	* aplikasi	* sudah oke * urus HKI - manual book - dokumen teknikal - apk - berkas HKI di p3m	
6.	27/06/2025	* laporan	* revisi laporan sesuai catatan	
7.	8/07/2025	* laporan	* revisi laporan sesuai catatan	

8	14/07 2025	* laporan	* acc laporan * pendaftaran ujian	

Tegal, 15 Juli 2025
Dosen Pembimbing II

Hepatika Zidny Ilmadina, S.Pd., M.Kom.
NIPY. 08.017.340

Lampiran 9 Responden SUS

No	Email	Nama	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1.	mutmainahresbbs@gmail.com	Mutmainah	5	1	4	2	4	2	5	1	5	1
2.	hasnitaranikumala@gmail.com	Rani Hasnita	5	1	5	1	5	1	5	1	5	1
3.	nurulfebyanisa26@gmail.com	Nurul Febi	5	1	5	1	5	1	5	1	5	1
4.	feyzhuu@gmail.com	Feyy	5	1	5	1	5	1	5	1	5	1
5.	annurriyadhus17@gmail.com	Annur	4	1	4	2	4	2	5	2	5	2
6.	fandhy890@gmail.com	Fandhy Satria Herlambang	5	1	5	1	5	1	5	1	5	1
7.	rakhmadhaninurulaini@gmail.com	Rakhmadhani Nurul Aini	4	2	4	3	4	2	4	2	4	3
8.	lloop1157@gmail.com	Faiz Fadhillah Umar	5	1	5	1	5	1	5	1	5	1
9.	fadilarizka2002@gmail.com	Dila	4	1	5	1	5	1	5	1	5	1
10.	putriajengimamah.study@gmail.com	Putri Ajeng Imamah	4	2	4	2	4	2	4	2	4	2
11.	dhiyashilmia@gmail.com	DHIYA SHILMI AULIA	4	3	5	3	4	3	3	2	5	3
12.	tpermatasari820@gmail.com	Tiara Permatasari	3	2	4	1	4	2	4	1	4	2
13.	dhiyaannabilah03@gmail.com	bila	4	2	5	3	5	2	5	1	5	5
14.	riyanbayu0102@gmail.com	Riyan Bayu saputra	4	4	4	4	4	4	4	4	4	4
15.	r.salsabilah.q@gmail.com	Shalsha Bilah	4	2	4	2	4	2	4	2	4	2
16.	ainulkharis04@gmail.com	Ainul Kharis	3	2	4	2	5	3	4	3	5	4
17.	saatirahkh@gmail.com	Saatirah Khoirunnisa	3	3	3	3	4	2	3	2	3	3
18.	bimaaryafatah23@gmail.com	Bima Arya Fatah	4	2	4	5	5	1	4	1	5	1
19.	erlinaputrii132@gmail.com	Erlina	3	3	3	4	4	2	4	2	5	1
20.	aisyifatia23@gmail.com	Rohadatul Aisyi Fatikha	5	2	5	5	5	1	5	1	5	3
21.	dindatheanekawati@gmail.com	Dinda Ari	4	2	4	2	4	2	4	1	5	2
22.	ikasukmawati005@gmail.com	Ika sukmawati	5	1	4	2	5	2	4	2	5	2
23.	arifatulfadia.1620@gmail.com	Balkis Arifatul Fadia	4	1	5	2	4	1	3	5	5	2

24.	putriiwillaandiinii1102@gmail.com	Putri Willa Andini	3	2	4	2	4	3	3	3	4	3
25.	syaharanibilla1@gmail.com	Syaharani Salsabilla Prasetyo	4	2	5	3	4	1	4	1	5	2
26.	atafarisa@gmail.com	Athafarisya	3	2	3	2	4	2	3	2	3	2