LAMPIRAN

Lampiran 1. Surat Kesepakatan Bimbingan Skripsi

SURAT KESEPAKATAN BIMBINGAN SKRIPSI

Kami yang bertanda tangan dibawah ini:

Pihak Pertama

Nama	: Firda Aulia Rakhmah
NIM	: 21090119
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

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

Pada hari ini Rabu tanggal 05 Maret 2025 telah terjadi sebuah kesepakatan bahwa Pihak Kedua bersedia menjadi Pembimbing I Skripsi Pihak Pertama dengan syarat Pihak Pertama wajib melakukan bimbingan Skripsi minimal 2 minggu sekali kepada Pihak Kedua. Adapun waktu dan tempat pelaksanaan disepakati antar pihak.

Demikian kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Skripsi

Tegal, 05 Maret 2025

Pihak Pertama	Pihak Kedua
Firda Aulia Rakhmah	Dyah Apriliani, 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	: Firda Aulia Rakhmah
NIM	: 21090119
Program Studi	: Sarjana Terapan Teknik Informatika

Pihak Kedua

Nama	: Hepatika Zidny Ilmadina, S.Pd., M.Kom.
Status	: Dosen
NIDN	: 0618119101
Jabatan Fungsional	: Asisten Ahli
Pangkat/Golongan	: Penata Muda Tingkat I / III-B

Pada hari ini Rabu tanggal 05 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, 05 Maret 2025

Pihak Pertama	Pihak Kedua
Firda Aulia Rakhmah	Hepatika Zidny Ilmadina, S.Pd., M.Kom.

Mengetahui

Ketua Program Studi Sarjana Terapan Teknik Informatika

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

NIPY. 09.015.225

Lampiran 2. Surat Pernyataan Pengajuan HKI

SURAT PERNYATAAN

Yang bertanda tangan di bawah ini, pemegang hak cipta:

1. Nama : Firda Aulia Rakhmah
   Kewarganegaraan : Indonesia
   Alamat : Desa Kertasinduyasa RT.03/ RW.03, Kecamatan Jatibarang, Kabupaten Brebes

2. Nama : Dyah Apriliani
   Kewarganegaraan : Indonesia
   Alamat : Perumahan Mutiara Indah Block C4, Jl. Nangka Gg.2, RT.02/ RW.02, Kelurahan Procot, Kecamatan Slawi, Kabupaten Tegal

3. Nama : Hepatika Zidny Ilmadina
   Kewarganegaraan : Indonesia
   Alamat : Jalan Kenanga Gang I/9, RT. 03/ RW. 01, Kelurahan Mangkukusuman, Kecamatan Tegal Timur, Kota Tegal

Dengan ini menyatakan bahwa:

1. Karya Cipta yang saya mohonkan:
   Berupa : Program Komputer
   Berjudul : Aplikasi Pendukung Kesehatan Mental Dengan AI *Real-Time Facial Emotion Recognition* Menggunakan Arsitektur *Long Short-Term Memory* (LSTM)

   - 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 sebagimana mestinya.

Tegal, 17 Juni 2025

(Firda Aulia Rakhmah)
Pemegang Hak Cipta*

(Dyah Apriliani)
Pemegang Hak Cipta*

(Hepatika Zidny Ilmadina)
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. Nama : Firda Aulia Rakhmah
   Kewarganegaraan : Indonesia
   Alamat : Desa Kertasinduyasa RT.03/ RW.03, Kecamatan Jatibarang, Kabupaten Brebes

2. Nama : Dyah Apriliani
   Kewarganegaraan : Indonesia
   Alamat : Perumahan Mutiara Indah Block C4, Jl. Nangka Gg.2, RT.02/ RW.02, Kelurahan Procot, Kecamatan Slawi, Kabupaten Tegal

3. Nama : Hepatika Zidny Ilmadina
   Kewarganegaraan : Indonesia
   Alamat : Jalan Kenanga Gang I/9, RT. 03/ RW. 01, Kelurahan Mangkukusuman, Kecamatan Tegal Timur, Kota Tegal

Adalah **Pihak I** selaku pencipta, dengan ini menyerahkan karya ciptaan saya kepada :
Nama : Pusat Penelitian dan Pengabdian Masyarakat (P3M)
Politeknik Harapan Bersama
Alamat : Jalan Mataram Nomor 9, Kelurahan Pesurungan Lor,
Kecamatan Margadana, Kota Tegal

Adalah **Pihak II** selaku Pemegang Hak Cipta berupa Program Komputer dengan judul "Aplikasi Pendukung Kesehatan Mental Dengan AI *Real-Time Facial Emotion Recognition* Menggunakan Arsitektur *Long Short-Term Memory* (LSTM)" 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, 17 Juni 2025

Pemegang Hak Cipta
Kepala P3M

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

Pencipta

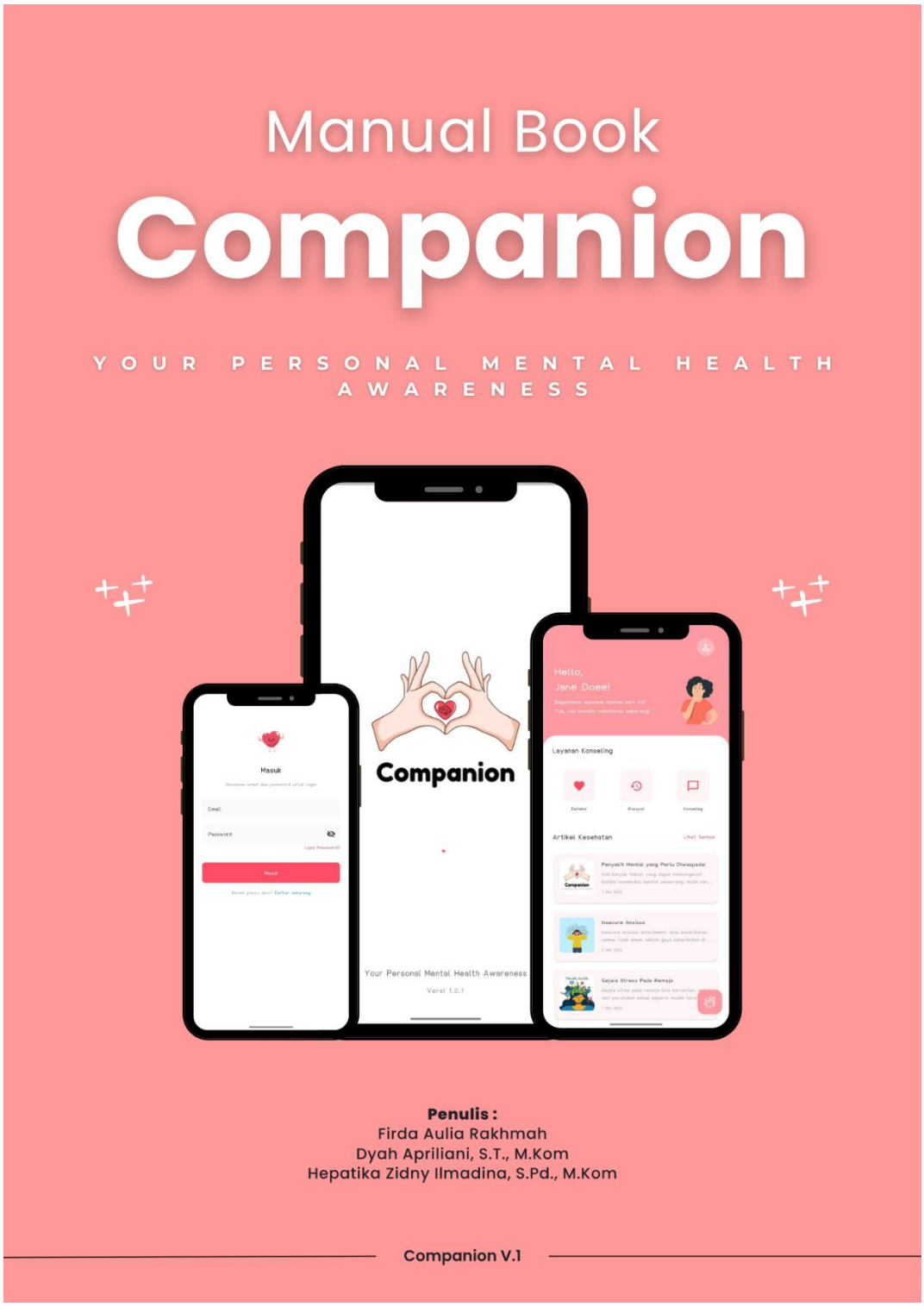
METERAI TEMPEL C1AKX555959098

(Firda Aulia Rakhmah)

(Dyah Apriliani)

(Hepatika Zidny Ilmadina)

Lampiran 4. Manual Book dan Dokumen Teknikal



Daftar Isi

A. Pendahuluan

Companion

YOUR PERSONAL MENTAL HEALTH AWARENESS

Selamat datang di Companion — Aplikasi pendamping kesehatan mental yang dirancang untuk membantu Anda mengenali, memahami, dan mengelola emosi secara lebih baik. Di tengah tekanan hidup yang modern, menjaga kesehatan mental menjadi kebutuhan yang tak kalah penting dari kesehatan fisik.

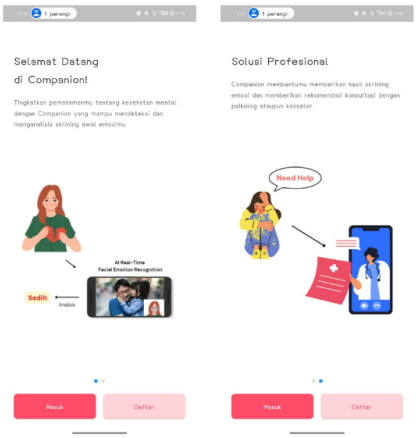
Companion hadir sebagai solusi cerdas yang menggabungkan teknologi kecerdasan buatan (Artificial Intelligence) dengan pendekatan psikologis untuk memberikan dukungan yang Anda butuhkan. Melalui fitur utama deteksi emosional secara realtime, Aplikasi ini dapat mengenali tingkat emosional dan memberikan hasil skrining awal yang relevan dengan kondisi kesehatan mental Anda. Selain itu, Companion juga menyediakan akses konseling profesional secara online yang membantu mendapatkan dukungan kapanpun sesuai kebutuhan Anda.

Aplikasi ini dirancang agar mudah digunakan oleh masyarakat umum usia dewasa muda 18-29 tahun, kapan pun, dan di mana pun. Dengan Companion, Anda tidak sendiri. Kami hadir sebagai sahabat digital yang siap menemani perjalanan Anda menuju kehidupan yang lebih seimbang, sehat, dan bermakna.

B. Tampilan Awal

Begitu Anda membuka Aplikasi Companion untuk pertama kalinya, Anda akan melihat layar pembuka (splash screen) yang menampilkan logo dan identitas aplikasi sebagai langkah awal untuk mendampingi perjalanan kesehatan mental Anda.

Halaman splash screen menampilkan logo dan tagline Companion sebagai sambutan awal bagi Anda. Setelah beberapa detik, aplikasi akan membawa Anda ke halaman onboarding yang memperkenalkan fitur utama dan bagaimana Companion dapat membantu Anda menjaga kesehatan mental.



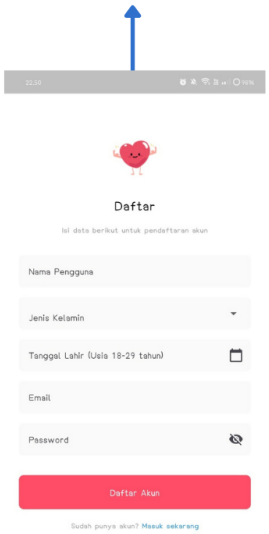
C. Panduan Pengguna

1. Registrasi dan Login

Sebelum mengakses aplikasi Companion, pastikan Anda sudah terhubung ke jaringan internet yang stabil. Aplikasi ini hanya ditujukan untuk pengguna berusia 18–29 tahun.

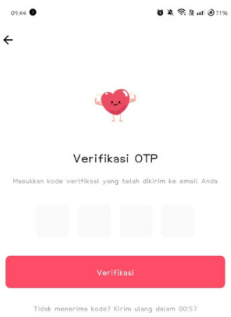
Langkah 1

Isi nama, jenis kelamin, tanggal lahir, email, dan password Anda untuk mendaftar.



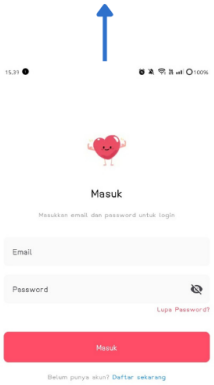
Langkah 2

Cek email Anda dan masukkan kode OTP yang telah dikirimkan untuk melanjutkan proses pendaftaran.



Langkah 3

Masuk ke aplikasi dengan email dan password yang sudah Anda daftarkan.



2. Beranda

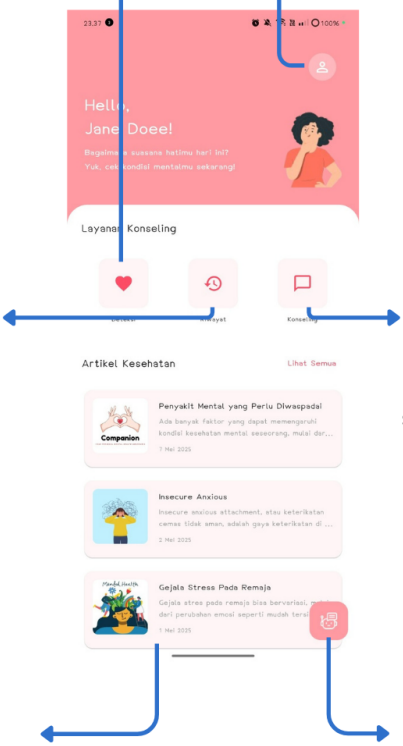
Setelah login berhasil, Anda akan masuk ke halaman Beranda. Di sini, Anda bisa langsung menikmati dan menggunakan berbagai fitur utama aplikasi Companion.

Deteksi Basic Emotion

Digunakan untuk mendeteksi tingkat emosional secara real-time berdasarkan ekspresi wajah Anda saat bercerita.

Profil Pengguna

Anda dapat mengelola dan memperbarui informasi akun, termasuk nama lengkap, email, dan password, agar data Anda selalu terkini dan aman.



Riwayat

Menampilkan riwayat skrining dan pembayaran Anda selama menggunakan aplikasi Companion untuk memudahkan pengelolaan aktivitas.

Konseling

Di sini Anda bisa melihat psikolog yang sudah Anda pesan dan bayar. Anda dapat memulai konsultasi tepat waktu sesuai jadwal yang telah ditentukan sebelum pembayaran.

Artikel Kesehatan

Temukan berbagai artikel seputar kesehatan mental yang bisa membantu Anda merawat dan meningkatkan kesejahteraan emosional setiap hari.

Chatbot

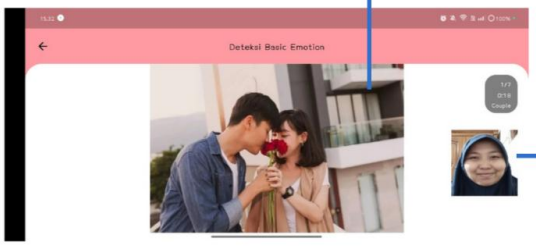
Chatbot siap membantu Anda kapan saja dengan jawaban cepat dan tips mudah tentang kesehatan mental.

3. Deteksi Basic Emotion

Sebelum memulai proses skrining awal pada fitur Deteksi Basic Emotion, Anda akan melihat dua pop-up notifikasi secara berurutan. Notifikasi ini bertujuan untuk memberikan panduan singkat agar proses skrining awal berjalan dengan nyaman dan lancar.

Anda akan diarahkan untuk memutar posisi ponsel ke mode landscape agar tampilan lebih optimal selama proses skrining berlangsung. Setelah Anda klik tombol "Selanjutnya", Anda akan diarahkan ke notifikasi ke-2.

Pada notifikasi ke-2 ini, Anda akan diminta untuk mengungkapkan perasaan berdasarkan gambar (stimulus) yang muncul setelah Anda menekan tombol "Mulai". Jawablah sesuai dengan perasaan Anda saat ini untuk mendapatkan hasil yang lebih akurat.



Langkah 1

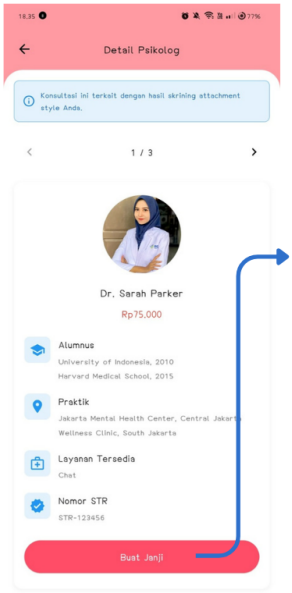
Anda akan melihat 14 gambar stimulus dengan tema yang berbeda. Setiap gambar akan ditampilkan selama 10 detik. Anda diminta untuk bercerita berdasarkan perasaan yang muncul.

Selama 2 menit 20 detik, sistem merekam ekspresi wajah Anda saat bercerita berdasarkan stimulus. Hasilnya akan menunjukkan: secure, insecure-anxious, atau insecure-avoidant.

Langkah 2

Hasil skrining akan ditampilkan setelah Anda bercerita dari 14 stimulus. Jika hasil menunjukkan tipe Insecure-anxious atau Insecure-avoidant, Anda dapat mengunduh resume hasil skrining untuk keperluan konsultasi di luar aplikasi dengan biaya tambahan.

Pada tahap ini juga, Anda juga dapat mengakses fitur Konseling, yang memungkinkan Anda berkonsultasi dengan psikolog sesuai kebutuhan.

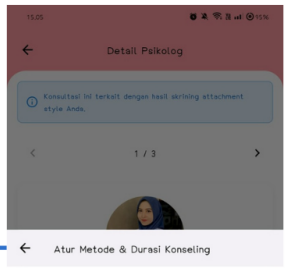


Langkah 3

Jika Anda memilih tombol "Konseling", akan tersedia beberapa pilihan psikolog yang disesuaikan dengan kebutuhan dan kenyamanan Anda. Anda juga dapat melihat informasi detail masing-masing psikolog sebelum memilih sesi konseling.

Langkah 4

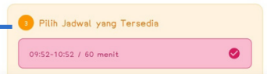
Di tahap ini, Anda bisa membuat janji temu dengan psikolog sesuai dengan kebutuhan dan waktu yang paling cocok untuk Anda.



Hari menyesuaikan dengan jadwal Anda.

Pilihlah hari sesuai tanggal yang tersedia.

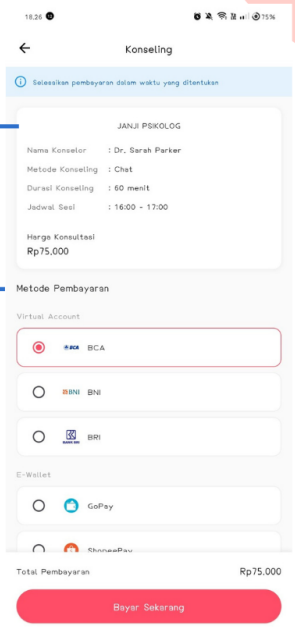
Jam konsultasi sesuai jadwal yang masih tersedia di sistem.



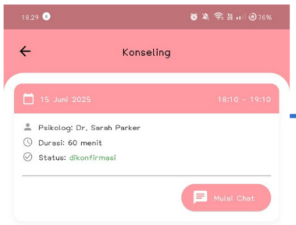
Klik "Lanjutkan ke Pembayaran" untuk membuat pesanan.

Langkah 5

Anda dapat menyelesaikan transaksi untuk jadwal konseling yang telah dibuat. Sebelum melakukan pembayaran, Anda dapat memastikan kembali nama psikolog, metode, durasi, dan jadwal konseling yang dipilih.



Tersedia berbagai metode pembayaran yang bisa Anda pilih. Silakan lakukan pembayaran sesuai dengan jumlah nominal yang tertera.

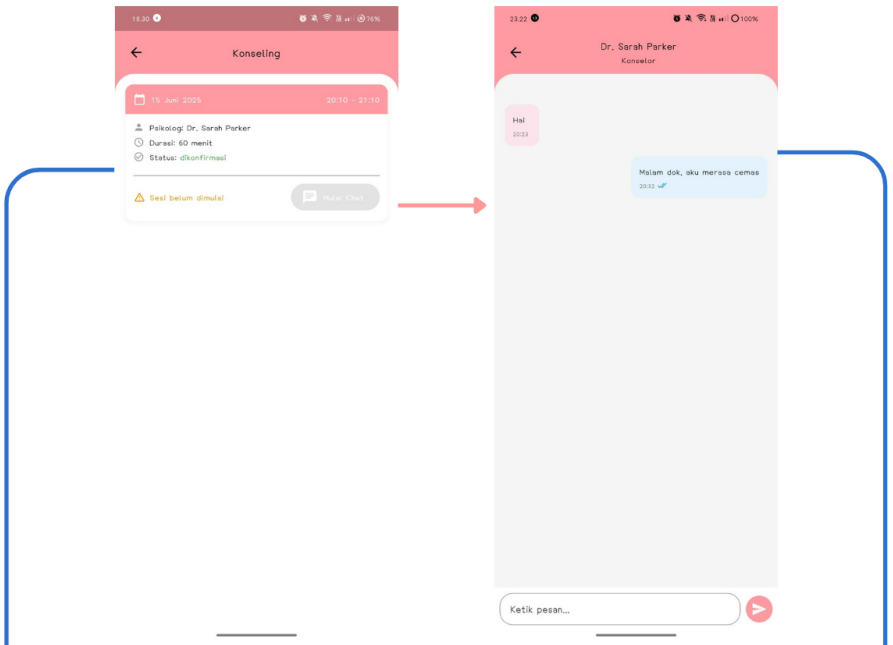


Langkah 6

Setelah menyelesaikan transaksi, Anda dapat mengakses layanan konseling dengan psikolog sesuai jadwal yang telah dipilih. Pada tahap ini, Anda dapat memulai sesi konseling sesuai dengan metode dan durasi yang telah dibayarkan.

4. Konseling

Halaman ini hanya menampilkan sesi bagi pengguna yang telah berhasil melakukan booking dan menyelesaikan pembayaran konseling.



Langkah 1

Setelah pembayaran sesi konseling berhasil, detail booking akan tampil di halaman Booking Aktif dalam fitur Konseling. Anda dapat melihat informasi sesi seperti jadwal, nama psikolog, durasi, jenis layanan, dan status booking. Saat mendekati waktu sesi, tombol "Mulai Chat" akan aktif dan siap digunakan.

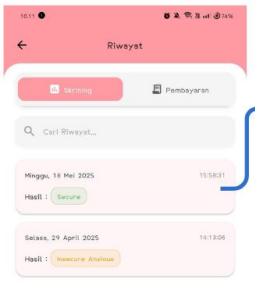
Langkah 2

Setelah menekan tombol "Mulai Chat", Anda akan diarahkan ke halaman chat pribadi dengan psikolog. Di sini, Anda dapat berkomunikasi secara real-time selama durasi sesi yang ditentukan. Antarmuka chat dibuat sederhana dengan pesan pengguna dan psikolog yang dibedakan warnanya.

5. Riwayat

Fitur Riwayat menampilkan rekam jejak Anda dalam dua menu, yaitu Riwayat Skrining untuk hasil deteksi basic emotion dan Riwayat Pembayaran untuk daftar transaksi. Riwayat skrining atau pembayaran ini hanya muncul jika Anda pernah melakukan skrining atau transaksi di aplikasi Companion.

- **Riwayat Skrining**



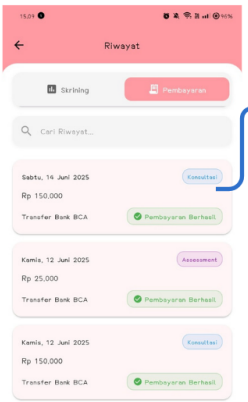
Langkah 1

Menampilkan daftar hasil deteksi emosional yang telah Anda lakukan, lengkap dengan tanggal, waktu pelaksanaan dan hasil skrining awalnya.

Langkah 2

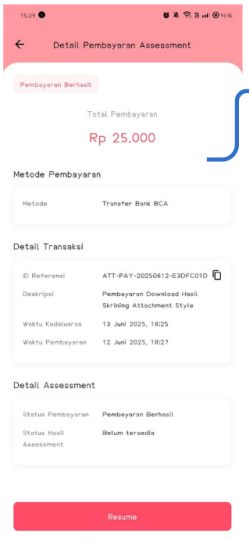
Jika Anda mengklik salah satu entri pada tab Riwayat Skrining, Anda akan diarahkan ke halaman hasil skrining yang menampilkan detail seperti saat Anda pertama kali melakukan deteksi basic emotion.

- **Riwayat Pembayaran**



Langkah 1

Menampilkan daftar pembayaran yang telah Anda lakukan, lengkap dengan tanggal dan status transaksi.



Langkah 2

Jika Anda mengklik salah satu entri pada tab Riwayat Pembayaran, Anda akan diarahkan ke halaman detail pembayaran yang menampilkan detail transaksi yang dilakukan.

6. Artikel Kesehatan

Anda dapat melihat daftar artikel kesehatan yang berfokus pada topik kesehatan mental, seperti pengelolaan emosi, stres, kecemasan, hubungan interpersonal, dan tips menjaga kesehatan mental sehari-hari. Setiap artikel dilengkapi dengan informasi penulis dan tanggal publikasi, sehingga Anda dapat memperoleh wawasan yang terpercaya dan relevan untuk menunjang kesehatan mental Anda.

Langkah 1

Klik "Lihat Semua" untuk membuka daftar artikel lengkap.

Langkah 2

Pilih artikel yang ingin dibaca.

Langkah 3

Artikel terbuka dalam tampilan penuh.

7. Chatbot

CnBuddy adalah chatbot interaktif yang membantu Anda memahami isu seputar kesehatan mental. Melalui fitur ini, Anda dapat mengajukan pertanyaan sederhana seperti "Apa itu kesehatan mental?", dan akan mendapatkan jawaban informatif secara otomatis. CnBuddy menyapa Anda dengan ramah, menanyakan kondisi Anda, dan memberikan penjelasan yang mudah dipahami. Fitur ini dirancang sebagai pendamping awal sebelum Anda melanjutkan ke layanan konseling dengan psikolog di aplikasi.

Langkah 1

Ketik pertanyaan yang ingin Anda tanyakan seperti "Apa itu kesehatan mental?" pada kolom input bagian bawah.

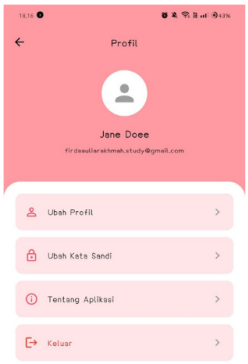
Langkah 2

CnBuddy akan merespons secara otomatis dengan jawaban yang informatif dan mudah dipahami.

Langkah 3

Lanjutkan percakapan sesuai kebutuhan Anda. Fitur ini dapat membantu mengenal kondisi mental secara umum sebelum berkonsultasi dengan psikolog.

8. Profil Pengguna



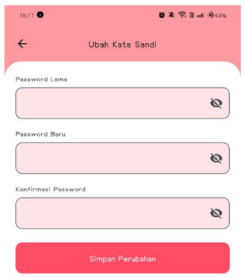
Di halaman Profil, Anda dapat melihat informasi dasar akun seperti nama dan email dengan tampilan yang simpel dan mudah digunakan.

Keluar

Ketuk tombol "Keluar" untuk keluar dari akun. Login kembali diperlukan untuk mengakses fitur aplikasi.

Ubah Profil

Ketuk "Ubah Profil" untuk mengedit nama dan email akun Anda.



Ubah Kata Sandi

Ketuk "Ubah Password" untuk mengganti kata sandi demi keamanan akun.

Tentang Aplikasi

Klik tombol "Tentang Aplikasi" untuk melihat informasi mengenai versi aplikasi, tim pengembang, tujuan pembuatan, serta kontak bantuan jika diperlukan.

D. Panduan Psikolog

1. Registrasi dan Login

Sebelum mengakses aplikasi Companion, pastikan Anda sebagai psikolog telah memenuhi beberapa ketentuan berikut:

- Pastikan perangkat Anda memiliki koneksi internet yang stabil untuk memastikan semua fitur aplikasi dapat berjalan dengan optimal.
- Anda harus sudah terdaftar sebagai ekspert (psikolog) di sistem Companion. Proses pendaftaran dilakukan melalui admin kami.
- Pastikan Anda telah menerima akun login dari admin. Jika belum, silakan hubungi tim admin kami untuk melakukan pendaftaran akun sebagai ekspert terlebih dahulu.

Langkah 1

Jika Anda sudah terdaftar dan memiliki akun, Anda dapat langsung melakukan login dengan mengisikan email dan password yang telah diberikan.

2. Beranda

Setelah login berhasil, Anda akan masuk ke halaman Beranda Psikolog. Di sini, peran Anda dimulai untuk mendampingi, memberi arahan, dan menjadi teman bagi mereka yang membutuhkan dukungan.

Konseling

Anda dapat memulai sesi konseling dengan pasien yang sudah melakukan pembayaran.

Jadwal

Di sini Anda dapat mengatur hari dan durasi sesi konseling sesuai dengan waktu dan ketersediaan Anda.

Janji Temu

Daftar semua sesi konsultasi yang dijadwalkan pada hari ini.

Statistik Pasien

Anda dapat melihat Riwayat pasien yang pernah melakukan konseling dengan anda dan uang pemasukan sehingga Anda selalu update dengan aktivitas praktik Anda.

Profil Psikolog

Anda dapat memperbarui informasi akun, seperti nama lengkap, email, dan password, agar data tetap terkini.

Antrean

Menampilkan daftar pasien yang akan melakukan sesi konseling dengan Anda, sehingga Anda dapat memantau jadwal dengan lebih mudah.

Riwayat

Daftar semua konsultasi yang telah selesai dilakukan.

Chatbot

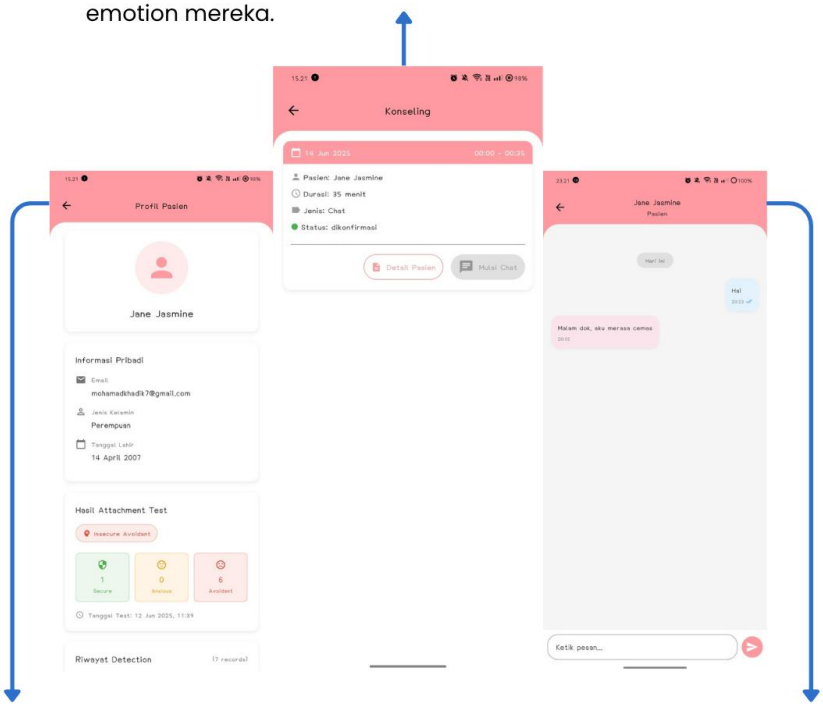
Chatbot siap membantu Anda kapan saja dengan jawaban cepat dan tips mudah tentang kesehatan mental.

3. Konseling

Padah fitur ini Anda dapat langsung melakukan sesi konsultasi dengan pasien yang sudah melakukan booking dan menyelesaikan pembayaran. Pastikan setiap sesi berjalan lancar untuk memberikan dukungan terbaik.

Langkah 1

Halaman Konseling menampilkan daftar pasien beserta waktu, durasi sesi, dan status pembayaran. Anda juga dapat melihat detail pasien dan riwayat deteksi basic emotion mereka.



Langkah 2

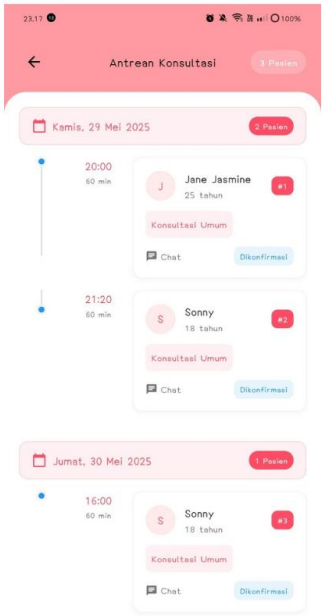
Saat Anda mengklik tombol "Detail Pasien", akan ditampilkan informasi identitas pasien serta riwayat hasil deteksi emosi yang pernah dilakukan.

Langkah 3

Klik tombol "Mulai Chat" untuk memulai sesi konsultasi secara real-time dengan pasien yang telah terjadwal.

4. Antrean

Fitur ini menampilkan semua pasien yang telah terjadwal untuk melakukan sesi konsultasi dengan Anda — baik untuk hari ini, besok, maupun hari-hari berikutnya.



Setiap sesi ditampilkan dalam urutan berdasarkan tanggal dan waktu, lengkap dengan informasi pasien seperti nama, usia, jenis layanan, serta status konfirmasi. Anda juga dapat melihat durasi setiap sesi dan mulai konsultasi saat waktunya tiba.

Tampilan disusun seperti garis waktu (timeline), sehingga memudahkan Anda untuk memantau dan mengelola seluruh antrean secara lebih rapi dan terjadwal.

5. Jadwal

Di halaman ini, Anda sebagai psikolog dapat dengan mudah mengatur hari dan waktu yang tersedia untuk melakukan sesi konsultasi bersama pengguna.

Langkah 1

Pada bagian atas halaman, Anda akan melihat informasi profil beserta jumlah total hari kerja yang sudah diatur. Setiap hari kerja akan ditampilkan dalam daftar, lengkap dengan jam konsultasi yang sudah ditambahkan.

Langkah 2

Untuk menambahkan hari kerja baru, tekan tombol "+ Tambah Hari" yang ada di bagian bawah. Anda bisa memilih hari apa saja yang Anda inginkan sesuai dengan waktu luang Anda untuk menerima sesi konsultasi.

Langkah 3

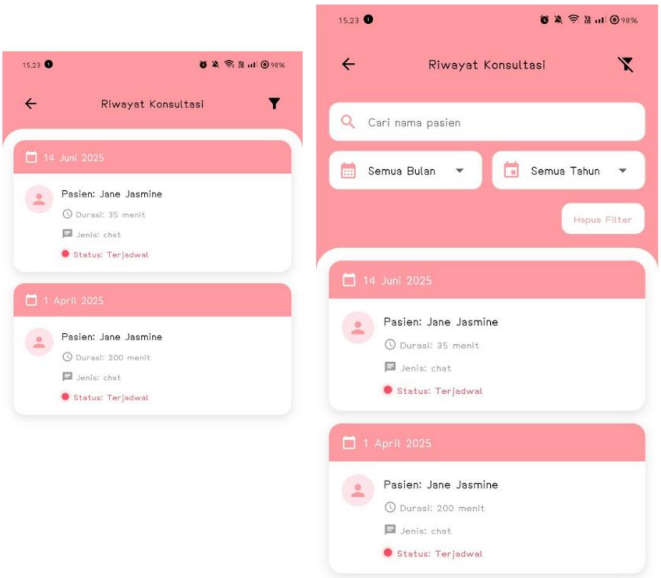
Setelah memilih hari, Anda bisa menambahkan satu atau beberapa jam konsultasi pada hari tersebut dengan menekan tombol "+ Tambah Jam." Atur waktu mulai dan waktu selesai sesuai keinginan Anda.

Langkah 4

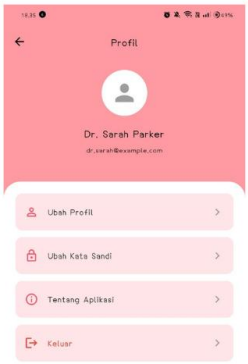
Jika Anda ingin mengubah waktu yang sudah ditentukan, tekan ikon edit ( ) di samping jadwal tersebut. Untuk menghapus jam atau hari kerja, tekan ikon hapus ( ) yang tersedia di setiap bagian.

6. Riwayat

Halaman Riwayat ini menampilkan riwayat konseling yang sudah selesai dilakukan oleh pengguna bersama Anda sebagai psikolog. Di dalam halaman ini, Anda dapat melihat informasi penting dari setiap sesi, seperti nama pengguna, durasi konseling, serta jenis layanan yang digunakan. Halaman ini juga dilengkapi dengan fitur filter yang memudahkan Anda untuk menyaring riwayat berdasarkan tanggal atau jenis layanan, sehingga Anda dapat dengan mudah meninjau kembali sesi-sesi yang telah berlangsung.



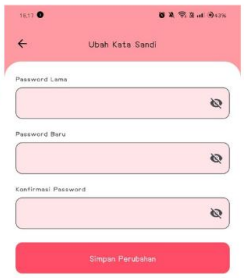
7. Profil Psikolog



Di halaman Profil, Anda dapat melihat informasi dasar akun seperti nama dan email dengan tampilan yang simpel dan mudah digunakan.

Keluar

Ketuk tombol "Keluar" untuk keluar dari akun. Login kembali diperlukan untuk mengakses fitur aplikasi.



Ubah Profil

Ketuk "Ubah Profil" untuk mengedit nama dan email akun Anda.

Ubah Kata Sandi

Ketuk "Ubah Password" untuk mengganti kata sandi demi keamanan akun.

Tentang Aplikasi

Klik tombol "Tentang Aplikasi" untuk melihat informasi mengenai versi aplikasi, tim pengembang, tujuan pembuatan, serta kontak bantuan jika diperlukan.

E. Troubleshooting

1. Gagal Login / Registrasi

Penyebab : Email sudah terdaftar, Format email salah, Batas usia tidak sesuai.
Solusi :

- Cek format email & password.
- Jika email sudah pernah digunakan, bisa menggunakan email lainnya.
- Coba login ulang setelah beberapa saat jika server sedang sibuk.
- Pastikan pengguna berada dalam range usia 18 - 29 tahun.

2. Deteksi Basic Emotion Tidak Bisa Diakses / Gagal

Penyebab : Kamera tidak diizinkan, Konekti internet tidak stabil.
Solusi :

- Pastikan izin kamera diaktifkan.
- Periksa koneksi internet.
- Restart aplikasi dan coba ulangi lagi.

3. Loading Terlalu Lama

Penyebab : Internet tidak stabil, Backend overload, ada bug pada aplikasi.
Solusi :

- Gunakan WiFi atau jaringan internat lainnya yang lebih stabil.
- Tutup aplikasi lalu buka kembali.
- Coba akses fitur lain untuk memastikan hanya 1 fitur yang bermasalah.

4. Aplikasi Tidak Merespon

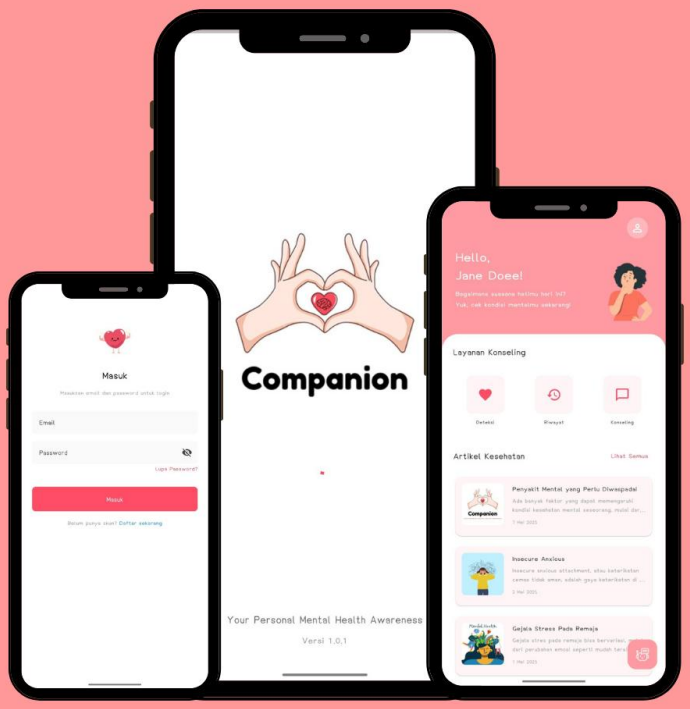
Penyebab : Ada update aplikasi.
Solusi :

- Coba tutup da buka kembali aplikasi Companion.
- Pastikan menggunakan versi terbaru dari aplikasi Companion.

Dokumen Teknis

Companion

YOUR PERSONAL MENTAL HEALTH AWARENESS



Penulis :
Firda Aulia Rakhmah
Dyah Apriliani, S.T., M.Kom
Hepatika Zidny Ilmadina, S.Pd., M.Kom

Companion V.1

Daftar Isi

A. Pendahuluan

Companion

YOUR PERSONAL MENTAL HEALTH AWARENESS

Selamat datang di Companion — Aplikasi pendamping kesehatan mental yang dirancang untuk membantu Anda mengenali, memahami, dan mengelola emosi secara lebih baik. Di tengah tekanan hidup yang modern, menjaga kesehatan mental menjadi kebutuhan yang tak kalah penting dari kesehatan fisik.

Companion hadir sebagai solusi cerdas yang menggabungkan teknologi kecerdasan buatan (Artificial Intelligence) dengan pendekatan psikologis untuk memberikan dukungan yang Anda butuhkan. Melalui fitur utama deteksi emosional secara realtime, Aplikasi ini dapat mengenali tingkat emosional dan memberikan hasil skrining awal yang relevan dengan kondisi kesehatan mental Anda. Selain itu, Companion juga menyediakan akses konseling profesional secara online yang membantu mendapatkan dukungan kapanpun sesuai kebutuhan Anda.

Aplikasi ini dirancang agar mudah digunakan oleh masyarakat umum usia dewasa muda 18-29 tahun, kapan pun, dan di mana pun. Dengan Companion, Anda tidak sendiri. Kami hadir sebagai sahabat digital yang siap menemani perjalanan Anda menuju kehidupan yang lebih seimbang, sehat, dan bermakna.

B. Spesifikasi Teknis

Spesifikasi teknis meliputi :

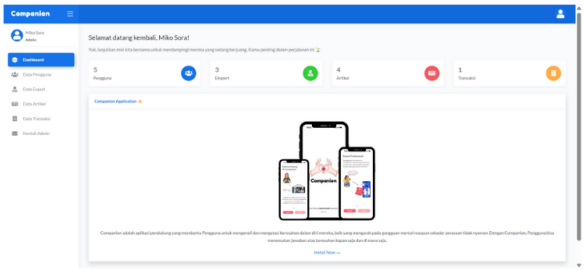
1. Modul Pengguna dan Psikolog
2. Source Code

Berikut uraian spesifikasi yang digunakan untuk membangun aplikasi :

1. Windows 11 Ram 14 Gb
2. Visual Studio Code
3. Web Browser
4. Smartphone

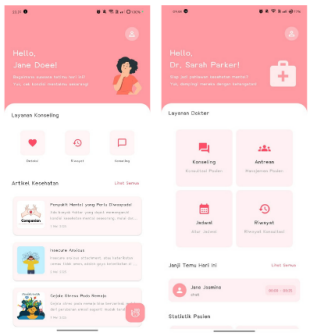
Berikut uraian spesifikasi modul :

1. Modul untuk developer



Untuk menjaga keamanan dan kendali sistem, halaman web ini dirancang khusus bagi developer guna melakukan pengelolaan data pengguna, konten, serta pemantauan aktivitas aplikasi

2. Modul untuk pengguna dan psikolog



Aplikasi Companion ini memiliki dua peran pengguna utama, yaitu masyarakat umum dan tenaga profesional seperti psikolog atau konselor. Setiap fitur dalam aplikasi disesuaikan dengan kebutuhan masing-masing peran.

C. Source Code

Source code backend dari aplikasi Companion dikembangkan menggunakan framework Flask-API. Backend ini menangani berbagai route penting seperti registrasi, login, dan lainnya. Selain itu, backend ini juga berfungsi sebagai website yang menampilkan halaman landing page untuk keperluan admin (developer) aplikasi.

1. Import Library

```python
import logging
import numpy as np
import traceback
import os
import pytz
import midtransclient
import shutil
import string
import tensorflow as tf
import random
import json
import mediapipe as mp
from app import create_app
from datetime import datetime
from flask import Flask
from flask import Blueprint, request, jsonify
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_jwt_extended import JWTManager
from flask_cors import CORS
from werkzeug.security import generate_password_hash, check_password_hash
from flask_jwt_extended import create_access_token, jwt_required, get_jwt_identity
from datetime import datetime, timedelta
from app.models import (User, Auth, Otp, db,HistoryDetection,psikolog,Education,PracticePlace,Payment,
                        WorkingHour,JadwalKonsultasi,AttachmentResult,WorkingDay)
from app.controllers.detection.video import ATTACHMENT_MAPPING,processor,determine_attachment_style,logger
from app.controllers.chatbot.chatbot import get_response,load_chatbot_resources,predict_class
from app.controllers.otp.index import send_otp_email,generate_otp
```

Script di atas adalah sebuah program Python yang mengimpor berbagai pustaka dan kerangka kerja seperti Flask, Flask-CORS, Flask-SQLAlchemy, dan lainnya. Library-Library ini digunakan untuk membangun bagian server (backend) dari sebuah aplikasi. Modul-modul yang digunakan mencakup:

- Flask API, yang digunakan untuk membuat dan menjalankan layanan web berbasis REST API.
- TensorFlow, yang memungkinkan pengembangan dan pemanfaatan model pembelajaran mesin (machine learning).
- Mediapipe, yang biasanya dimanfaatkan untuk mendeteksi titik-titik acuan (landmark) pada sebuah gambar atau video.

Script ini menyediakan berbagai fungsi dan kelas yang diperlukan untuk membangun logika server, mengelola permintaan API dari client, memproses data menggunakan model Machine Learning, serta melakukan analisis berbasis visual yang diintegrasikan menggunakan MediaPipe.

2. Konfigurasi Database

```python
BASE_DIR = Path(__file__).resolve().parent.parent
class Config:
    # Database configuration
    SQLALCHEMY_DATABASE_URI = os.getenv(
        "DATABASE_URI",
        "mysql+mysqlconnector://root:@localhost:3306/companion"
    )
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    # In your config.py
    STORAGE_PATH = BASE_DIR / 'storage'
    VIDEO_STORAGE = STORAGE_PATH / 'videos'

    # Create directories if they don't exist
    VIDEO_STORAGE.mkdir(parents=True, exist_ok=True)

    # JWT configuration
    JWT_SECRET_KEY ='your-secret-key-here'  # Change this in production!
    JWT_ACCESS_TOKEN_EXPIRES = False  # Token will not expire

    # Batasan upload
    MAX_CONTENT_LENGTH = 100 * 1024 * 1024  # 100MB
    ALLOWED_VIDEO_EXTENSIONS = {'mp4', 'mov', 'avi', 'mkv'}
    # Flask configuration
    SECRET_KEY = os.getenv('SECRET_KEY', 'your-secret-key-here')  # Change this in production!

```

Kode di atas merupakan bagian dari konfigurasi utama untuk aplikasi Flask. Konfigurasi ini terdiri dari tiga bagian, yaitu pengaturan basis data MySQL, pengelolaan token autentikasi JWT, dan pengamanan aplikasi dengan secret key. Pada bagian konfigurasi basis data, SQLALCHEMY_DATABASE_URI diatur untuk menyambungkan aplikasi ke database MySQL menggunakan driver mysqlconnector. Nilai URI ini bisa diambil dari environment variable DATABASE_URI, atau menggunakan default lokal. Pelacakan perubahan objek dinonaktifkan melalui SQLALCHEMY_TRACK_MODIFICATIONS untuk meningkatkan performa.

Untuk sistem autentikasi, digunakan JWT (JSON Web Token) dengan JWT_SECRET_KEY sebagai kunci utama untuk mengenkripsi token. Token ini tidak memiliki batas waktu kadaluarsa (JWT_ACCESS_TOKEN_EXPIRES = False), meskipun di sistem produksi, pengaturan ini disarankan untuk disesuaikan demi alasan keamanan. SECRET_KEY digunakan oleh Flask untuk mengamankan sesi pengguna dan fitur internal seperti CSRF protection. Nilai ini juga disarankan disimpan di environment variable agar tidak terekspos dalam kode sumber.

3. Inisialisasi dan Autentikasi SMTP

```
server = smtplib.SMTP('smtp.gmail.com', 587)
        server.starttls()
        server.login(sender_email, sender_password)
        server.send_message(msg)
        server.quit()
        logging.info(f"OTP sent to {email}")
        return True
```

Kode di atas digunakan untuk mengirim OTP (One-Time Password) ke email melalui server SMTP Gmail. Proses dimulai dengan membuat koneksi ke server smtp.gmail.com pada port 587, yang merupakan port standar untuk koneksi SMTP dengan protokol TLS.

Selanjutnya, koneksi diamankan menggunakan metode starttls() untuk mengaktifkan enkripsi TLS. Setelah itu, proses autentikasi dilakukan dengan memanggil login() menggunakan alamat email pengirim (sender_email) dan kata sandi (sender_password). Email yang akan dikirim didefinisikan dalam objek msg, dan dikirim melalui send_message(msg).

Setelah proses pengiriman selesai, koneksi ditutup menggunakan quit(). Logging digunakan untuk mencatat bahwa OTP telah berhasil dikirim ke alamat email tujuan, dan fungsi akan mengembalikan True sebagai tanda keberhasilan.

4. Route Register

```
@auth_bp.route('/register', methods=['POST'])
def register():
    data = request.get_json()

    # Validate required fields
    required_fields = ['name', 'email', 'password', 'gender_id']
    if not all(field in data for field in required_fields):
        logging.warning("Missing required fields in registration data")
        return jsonify({'message': 'Missing required fields'}), 400

    # Check if user already exists
    existing_user = User.query.filter_by(email=data['email']).first()

    # If user exists but hasn't verified OTP, allow re-registration
    if existing_user:
        if existing_user.verify_otp:
            logging.warning(f"Attempted registration with already verified email: {data['email']}")
            return jsonify({'message': 'Email already registered and verified'}), 400
        else:
            # Delete the unverified user record to allow re-registration
            try:
                # Delete any existing OTP records for this user
                Otp.query.filter_by(user_id=existing_user.id).delete()
                db.session.delete(existing_user)
                db.session.commit()
                logging.info(f"Deleted unverified user account for re-registration: {data['email']}")
            except Exception as e:
                db.session.rollback()
                logging.error(f"Failed to delete unverified user: {str(e)}")
                return jsonify({'message': 'Error processing registration'}), 500

```

```python
    # Create new user
    new_user = User(
        name=data['name'],
        email=data['email'],
        gender_id=data['gender_id'],
        tanggal_lahir=data.get('tanggal_lahir'),
        password=generate_password_hash(data['password']),
        verify_otp=False
    )

    try:
        db.session.add(new_user)
        db.session.commit()
        logging.info(f"New user registered: {new_user.name} with email: {new_user.email}")

        # Generate and send OTP
        otp = generate_otp()
        otp_expiry = datetime.now(local_timezone) + timedelta(minutes=5)

        new_otp = Otp(
            user_id=new_user.id,
            otp=generate_password_hash(otp),
            otp_expired_at=otp_expiry
        )

        db.session.add(new_otp)
        db.session.commit()
        logging.info(f"OTP generated and saved for user ID: {new_user.id}")

        # Send OTP via email
        if send_otp_email(new_user.email, otp):
            return jsonify({
                'message': 'Registration successful. Please verify your email with the OTP sent.',
                'user_id': new_user.id
            }), 201
        else:
            logging.error("Failed to send OTP email during registration")
            return jsonify({'message': 'Failed to send OTP email'}), 500

    except Exception as e:
        db.session.rollback()
        logging.error(f"Registration failed for email {data['email']}: {str(e)}")
        return jsonify({'message': f'Registration failed: {str(e)}'}), 500
```

Kode di atas mendefinisikan endpoint API untuk registrasi pengguna baru menggunakan metode POST melalui fungsi register. Fungsi ini menerima data dalam format JSON dari sisi klien, yang mencakup nama, email, password, dan gender sebagai data wajib. Jika ada data yang tidak lengkap, sistem akan mengembalikan respons kesalahan dengan status 400. Selanjutnya, sistem memeriksa apakah email sudah digunakan. Jika email sudah terdaftar dan telah diverifikasi sebelumnya, maka pendaftaran ditolak. Namun, jika akun dengan email tersebut belum diverifikasi, sistem akan menghapus akun dan data OTP lama agar pengguna dapat melakukan registrasi ulang.

Setelah validasi selesai, sistem membuat akun pengguna baru dan menyimpan kata sandi dalam bentuk hash untuk alasan keamanan. Kemudian, kode OTP acak akan dibuat, disimpan di database bersama waktu kedaluwarsanya, dan dikirim ke email pengguna melalui fungsi send_otp_email. Jika pengiriman email berhasil, sistem akan memberikan respons sukses bahwa registrasi berhasil dan pengguna diminta memverifikasi emailnya. Proses ini juga dilengkapi dengan logging aktivitas dan penanganan kesalahan untuk memastikan sistem berjalan dengan andal dan aman.

5. Route Login

```python
@auth_bp.route('/login', methods=['POST'])
def login():
    data = request.get_json()

    if not all(key in data for key in ['email', 'password']):
        logging.warning("Missing email or password during login")
        return jsonify({'message': 'Missing email or password'}), 400

    user = User.query.filter_by(email=data['email']).first()

    if not user or not check_password_hash(user.password, data['password']):
        logging.warning(f"Invalid login attempt for email: {data['email']}")
        return jsonify({'message': 'Invalid email or password'}), 401

    if not user.verify_otp:
        logging.warning(f"Unverified email login attempt: {user.email}")
        return jsonify({'message': 'Please verify your email first'}), 403

    # Create token with string ID
    access_token = create_access_token(identity=str(user.id))

    auth_record = Auth(
        user_id=user.id,
        login_at=datetime.now(local_timezone),
        token=access_token,
        token_expired_at=datetime.now(local_timezone) + timedelta(days=1)
    )

    try:
        db.session.add(auth_record)
        db.session.commit()
        logging.info(f"User logged in: {user.email}")
        return jsonify({
            'message': 'Login successful',
            'access_token': access_token,
            'user': {
                'id': user.id,
                'name': user.name,
                'email': user.email,
                'gender_id': user.gender_id,
                'role':user.role
            }
        }), 200

    except Exception as e:
        db.session.rollback()
        logging.error(f"Login failed for user ID {user.id}: {str(e)}")
        return jsonify({'message': f'Login failed: {str(e)}'}), 500
```

Kode sebelumnya mendefinisikan endpoint API untuk proses login pengguna menggunakan metode POST, yang menerima data JSON berisi email dan password. Jika salah satu tidak tersedia, sistem langsung merespons dengan status 400. Setelah itu, sistem mencari data pengguna berdasarkan email, jika tidak ditemukan atau password tidak cocok (diperiksa dengan check_password_hash()), maka login ditolak dengan status 401 dan dicatat sebagai upaya gagal. Jika pengguna ditemukan, sistem juga memverifikasi status akun, memastikan bahwa pengguna telah menyelesaikan verifikasi OTP —jika belum, login ditolak dengan status 403 dan pengguna diminta untuk memverifikasi email terlebih dahulu. Jika seluruh validasi lolos, sistem membuat token autentikasi JWT berdurasi satu hari menggunakan create_access_token() dengan user.id sebagai identitas. Token ini kemudian disimpan ke tabel Auth bersama waktu login dan kedaluwarsa. Bila penyimpanan token berhasil, sistem mengembalikan respons sukses (status 200) yang mencakup token dan detail pengguna seperti ID, nama, email, ID gender, dan peran. Jika terjadi kegagalan saat menyimpan token ke database, sistem melakukan rollback dan mencatat error untuk keperluan debugging.

6. Route Verifikasi Kode OTP

```python
@auth_bp.route('/verify-otp', methods=['POST'])
def verify_otp():
    data = request.get_json()

    if not all(key in data for key in ['user_id', 'otp']):
        logging.warning("Missing user_id or OTP in verification data")
        return jsonify({'message': 'Missing required fields'}), 400

    user = User.query.get(data['user_id'])
    if not user:
        logging.warning(f"User not found during OTP verification: ID {data['user_id']}")
        return jsonify({'message': 'User not found'}), 404

    # Get the latest OTP record
    otp_record = Otp.query.filter_by(user_id=user.id)\
        .order_by(Otp.created_at.desc()).first()

    if not otp_record:
        logging.warning(f"No OTP found for user ID {user.id}")
        return jsonify({'message': 'No OTP found'}), 404

    current_time = datetime.now(local_timezone)
    otp_expiry = otp_record.otp_expired_at
    if otp_expiry.tzinfo is None:
        otp_expiry = local_timezone.localize(otp_expiry)

    # Check if OTP is expired
    if current_time > otp_expiry:
        try:
            Otp.query.filter_by(user_id=user.id).delete()
            Auth.query.filter_by(user_id=user.id).delete()
            db.session.delete(user)
            db.session.commit()
            logging.warning(f"OTP expired. Account deleted for user ID {user.id}")
            return jsonify({'message': 'OTP has expired. Account has been deleted.'}), 400
        except Exception as e:
            db.session.rollback()
            logging.error(f"Error deleting account for expired OTP: {str(e)}")
            return jsonify({'message': f'Error deleting account: {str(e)}'}), 500

    # Verify OTP
    if check_password_hash(otp_record.otp, data['otp']):
        user.verify_otp = True
        db.session.commit()
        logging.info(f"Email verified for user ID {user.id}")
        return jsonify({'message': 'Email verified successfully'}), 200

    logging.warning(f"Invalid OTP attempt for user ID {user.id}")
    return jsonify({'message': 'Invalid OTP'}), 400
```

Kode sebelumnya adalah endpoint API /verify-otp yang digunakan untuk memverifikasi kode OTP saat registrasi atau lupa kata sandi. Sistem menerima user_id dan otp, lalu mencari data pengguna dan OTP terbaru di database. Jika OTP tidak ditemukan atau telah kedaluwarsa, akun pengguna akan dihapus untuk menjaga keamanan data dan mencegah penyalahgunaan. Jika OTP masih berlaku dan cocok dengan yang diinput, status verifikasi pengguna akan diaktifkan (verify_otp = True) dan sistem mengembalikan pesan sukses.

Seluruh proses ini dilengkapi dengan validasi input, pengecekan waktu kedaluwarsa, serta logging untuk keperluan audit dan pemantauan sistem. Prosedur ini penting untuk memastikan bahwa hanya pengguna yang benar-benar memiliki akses ke email terkait yang dapat mengaktifkan akun mereka, sehingga menjaga integritas dan keaslian data pengguna dalam sistem.

7. Route Lupa Password

```python
@auth_bp.route("/forgot-password/send-otp", methods=["POST"])
def forgot_password_send_otp():
    try:
        data = request.get_json()
        email = data.get("email")

        if not email:
            return jsonify({"success": False, "message": "Email diperlukan"}), 400

        # <--- PENTING: Validasi format email
        if not re.match(r"[^@]+@[^@]+\.[^@]+", email):
            return jsonify({"success": False, "message": "Format email tidak valid"}), 400

        user = User.query.filter_by(email=email).first()

        if not user:
            logging.warning(f"Attempt to send OTP for non-existent email: {email}")
            return jsonify({"success": True, "message": "Jika email terdaftar, kode OTP akan dikirim."}), 200

        otp_code = generate_otp() # Ini adalah OTP plain-text yang akan dikirim via email
        otp_expiry_time = datetime.now(local_timezone) + timedelta(minutes=5)

        # <--- PENTING: OTP DI-HASH SEBELUM DISIMPAN KE DATABASE
        hashed_otp = generate_password_hash(otp_code)

        new_otp = Otp(user_id=user.id, otp=hashed_otp, otp_expired_at=otp_expiry_time) # Menyimpan OTP yang sudah di-hash
        db.session.add(new_otp)
        db.session.commit()

        if send_otp_email(user.email, otp_code): # Kirim OTP plain-text via email
            logging.info(f"OTP for forgot password sent to {user.email} (user ID: {user.id})")
            return jsonify({
                "success": True,
                "message": "Kode OTP telah dikirim ke email Anda.",
                "user_id": user.id # SANGAT PENTING: Kirim user_id ini kembali ke Flutter!
            }), 200
        else:
            logging.error(f"Failed to send forgot password OTP email to {user.email}")
            return jsonify({"success": False, "message": "Gagal mengirim OTP. Silakan coba lagi."}), 500

    except Exception as e:
        db.session.rollback()
        logging.error(f"Error during forgot password OTP request: {str(e)}")
        logging.error(traceback.format_exc())
        return jsonify({"message": "Terjadi kesalahan server", "error": str(e)}), 500
```

Kode sebelumnya mendefinisikan endpoint API untuk menangani permintaan pengiriman OTP saat pengguna lupa kata sandi. Endpoint menerima input email dalam format JSON dan memvalidasi keberadaan serta format email; jika tidak valid, sistem mengembalikan pesan error. Terlepas dari apakah email ditemukan di database atau tidak, sistem tetap merespons sukses demi menjaga kerahasiaan data pengguna. Jika pengguna ditemukan, sistem menghasilkan OTP acak, menyimpannya dalam bentuk hash dengan masa berlaku 5 menit, lalu mengirimkan OTP asli ke email pengguna. Jika pengiriman berhasil, sistem merespons dengan pesan sukses dan user_id untuk proses verifikasi berikutnya. Seluruh proses dilengkapi dengan logging dan penanganan error agar sistem tetap andal dan aman.

8. Route Profile

```python
@auth_bp.route('/profile', methods=['GET'])
@jwt_required()
def get_profile():
    """Get user profile information including gender"""
    try:
        current_user_id = int(get_jwt_identity())
        logging.info(f"Fetching profile for user ID: {current_user_id}")

        user = User.query.get(current_user_id)
        if not user:
            logging.warning(f"Profile request for non-existent user ID: {current_user_id}")
            return jsonify({'message': 'User not found'}), 404

        gender_name = user.gender.gender if user.gender else None

        logging.info(f"Profile retrieved for user ID: {current_user_id}")
        return jsonify({
            'user': {
                'id': user.id,
                'name': user.name,
                'email': user.email,
                'gender_id': user.gender_id,
                'gender': gender_name,
                'tanggal_lahir': user.tanggal_lahir.isoformat() if user.tanggal_lahir else None  # Add date of birth
            }
        }), 200

    except Exception as e:
        logging.error(f"Error retrieving profile for user ID: {current_user_id}: {str(e)}")
        logging.debug("Traceback:\n" + traceback.format_exc())
        return jsonify({'message': f'Error retrieving profile: {str(e)}'}), 500
```

Endpoint /profile menggunakan metode GET untuk mengambil data profil pengguna yang sedang login, termasuk nama, email, jenis kelamin, dan tanggal lahir, serta dilindungi oleh @jwt_required() agar hanya dapat diakses oleh pengguna dengan token JWT yang valid. Setelah menerima permintaan, sistem mengambil user_id dari token menggunakan get_jwt_identity() dan mencari data pengguna melalui User.query.get(). Jika pengguna tidak ditemukan, sistem mengembalikan respons 404 dan mencatat log peringatan. Jika ditemukan, sistem juga mengambil nama gender dari relasi tabel Gender dan mengonversi tanggal lahir ke format ISO jika tersedia. Data profil kemudian dikemas dalam format JSON dan dikembalikan ke klien dengan status 200. Seluruh proses dilengkapi dengan logging untuk pemantauan serta penanganan error melalui blok try-except untuk menjaga kestabilan sistem.

9. Route Ubah Profile

```python
@auth_bp.route('/update-profile', methods=['PUT'])
@jwt_required()
def update_profile():
    current_user_id = get_jwt_identity()
    data = request.get_json()

    try:
        logging.info(f"Attempting to update profile for user ID: {current_user_id}")

        user = User.query.get(current_user_id)
        if not user:
            logging.warning(f"Profile update attempt for non-existent user ID: {current_user_id}")
            return jsonify({'message': 'User not found'}), 404

        email_updated = False
        if 'email' in data and data['email'] != user.email:
            existing_user = User.query.filter_by(email=data['email']).first()
            if existing_user and existing_user.id != int(current_user_id):
                logging.warning(f"Email update failed: {data['email']} already in use")
                return jsonify({'message': 'Email already registered to another account'}), 400

            logging.info(f"Updating email for user ID {current_user_id}: {data['email']}")
            user.email = data['email']
            user.verify_otp = False
            email_updated = True

            otp = generate_otp()
            otp_expiry = datetime.now(local_timezone) + timedelta(minutes=5)

            new_otp = Otp(
                user_id=user.id,
                otp=generate_password_hash(otp),
                otp_expired_at=otp_expiry
            )

            db.session.add(new_otp)
            send_otp_email(user.email, otp)
            logging.info(f"New OTP sent for email verification: user ID {current_user_id}")

        if 'name' in data:
            logging.info(f"Updating name for user ID {current_user_id}: {data['name']}")
            user.name = data['name']
        if 'gender_id' in data:
            logging.info(f"Updating gender_id for user ID {current_user_id}: {data['gender_id']}")
            user.gender_id = data['gender_id']
        if 'tanggal_lahir' in data:
            logging.info(f"Updating tanggal_lahir for user ID {current_user_id}: {data['tanggal_lahir']}")
            user.tanggal_lahir = data['tanggal_lahir']

        db.session.commit()
        logging.info(f"Profile updated successfully for user ID: {current_user_id}")

        response = {
            'message': 'Profile updated successfully',
            'user': {
                'id': user.id,
                'name': user.name,
                'email': user.email,
                'gender_id': user.gender_id,
                'gender': user.gender.gender if user.gender else None,
                'tanggal_lahir': user.tanggal_lahir.isoformat() if user.tanggal_lahir else None,
                'verify_otp': user.verify_otp
            }
        }

        if email_updated:
            response['message'] = 'Profile updated. Please verify your new email with the OTP sent.'

        return jsonify(response), 200

    except Exception as e:
        db.session.rollback()
        logging.error(f"Error updating profile for user ID {current_user_id}: {str(e)}")
        logging.debug("Traceback:\n" + traceback.format_exc())
        return jsonify({'message': f'Error updating profile: {str(e)}'}), 500
```

Endpoint update_profile pada route /update-profile memungkinkan pengguna yang sudah login (dengan JWT) untuk memperbarui data profil seperti nama, email, gender, dan tanggal lahir. Jika email diubah, sistem akan memastikan email baru belum digunakan pengguna lain, lalu mengirim OTP ke email tersebut untuk verifikasi. Data yang diubah disimpan ke database, dan jika email diperbarui, respons akan meminta pengguna untuk memverifikasi email baru. Proses ini mencakup log aktivitas dan penanganan kesalahan dengan rollback database jika terjadi error.

10. Route Ubah Kata Sandi

```python
@auth_bp.route('/change-password', methods=['PUT'])
@jwt_required()
def change_password():
    current_user_id = get_jwt_identity()
    data = request.get_json()

    try:
        logging.info(f"Password change requested for user ID: {current_user_id}")

        user = User.query.get(current_user_id)
        if not user:
            logging.warning(f"Password change attempt for non-existent user ID: {current_user_id}")
            return jsonify({'message': 'User not found'}), 404

        if 'current_password' not in data or 'new_password' not in data:
            logging.warning(f"Missing password fields for user ID: {current_user_id}")
            return jsonify({'message': 'Current and new password are required'}), 400

        if not check_password_hash(user.password, data['current_password']):
            logging.warning(f"Invalid current password for user ID: {current_user_id}")
            return jsonify({'message': 'Current password is incorrect'}), 401

        user.password = generate_password_hash(data['new_password'])

        Auth.query.filter_by(user_id=current_user_id).delete()

        new_token = create_access_token(identity=str(current_user_id))
        auth_record = Auth(
            user_id=current_user_id,
            login_at=datetime.now(local_timezone),
            token=new_token,
            token_expired_at=datetime.now(local_timezone) + timedelta(days=1)
        )
        db.session.add(auth_record)

        db.session.commit()
        logging.info(f"Password changed successfully for user ID: {current_user_id}")

        return jsonify({
            'success': True,
            'message': 'Password updated successfully',
            'access_token': new_token
        }), 200

    except Exception as e:
        db.session.rollback()
        logging.error(f"Error changing password for user ID {current_user_id}: {str(e)}")
        logging.debug("Traceback:\n" + traceback.format_exc())
        return jsonify({'message': f'Error changing password: {str(e)}'}), 500
```

Endpoint change_password pada endpoint /change-password digunakan untuk mengganti password pengguna yang sudah login menggunakan JWT. Sistem memverifikasi apakah current_password sesuai dengan password di database, lalu memperbaruinya dengan new_password yang sudah di-hash. Setelah perubahan, seluruh token login sebelumnya dihapus, dan token akses baru dibuat serta disimpan bersama waktu login dan masa berlakunya. Jika berhasil, sistem mengembalikan pesan sukses beserta token baru. Fungsi ini juga dilengkapi logging dan rollback jika terjadi kesalahan saat proses.

11. Route Artikel Kesehatan

```python
@artikel_bp.route('/artikel', methods=['GET'])
@jwt_required()
def get_artikel_list():
    """Ambil semua data artikel kesehatan"""
    try:
        current_user_id = int(get_jwt_identity())

        articles = ArtikelKesehatan.query.order_by(ArtikelKesehatan.tanggal_publikasi.desc()).all()
        result = []
        for a in articles:
            result.append({
                'id': a.id,
                'judul': a.judul,
                'penulis': a.penulis,
                'isi': a.isi,
                'tanggal_publikasi': a.tanggal_publikasi.strftime('%Y-%m-%d'),
                'gambar': url_for('static', filename=f'atlantis/assets/img/artikel/{a.gambar}', _external=True)
            })

        return jsonify({'articles': result}), 200

    except Exception as e:
        logging.error(f"Error fetching articles: {str(e)}")
        logging.debug("Traceback:\n" + traceback.format_exc())
        return jsonify({'message': f'Error fetching articles: {str(e)}'}), 500
```

Endpoint /artikel digunakan untuk mengambil daftar artikel kesehatan dari database dan mengirimkannya dalam format JSON melalui metode GET. Setelah autentikasi JWT diverifikasi dengan @jwt_required(), sistem mengambil user_id dari token, meskipun tidak digunakan lebih lanjut. Selanjutnya, sistem menjalankan query ke tabel ArtikelKesehatan menggunakan SQLAlchemy untuk mengambil semua data artikel, yang kemudian diurutkan berdasarkan tanggal publikasi terbaru. Setiap artikel diproses menjadi dictionary berisi id, judul, penulis, isi, tanggal publikasi, dan gambar, di mana URL gambar dibentuk lengkap menggunakan fungsi url_for() Flask. Hasilnya dikemas dalam JSON dan dikirim sebagai respons menggunakan jsonify(). Jika terjadi kesalahan saat pengambilan atau pemrosesan data, sistem mencatat error dan merespons dengan status 500.

12. Route Payment

```python
@payment_bp.route('/create-payment/<int:konsultasi_id>', methods=['POST'])
@jwt_required()
def create_payment(konsultasi_id):
    try:
        # Dapatkan user_id dari JWT token
        current_user_id = get_jwt_identity()

        # Get consultation data
        konsultasi = JadwalKonsultasi.query.get_or_404(konsultasi_id)

        # Get payment details from request
        data = request.get_json()
        payment_method = data.get('payment_method')
        payment_provider = data.get('payment_provider')

        # First create the payment record in your database
        payment = Payment.create_payment(
            konsultasi_id=konsultasi_id,
            user_id=current_user_id,  # Gunakan user_id dari token
            amount=konsultasi.biaya_konsultasi,
            payment_method=payment_method,
            payment_provider=payment_provider,
            va_number=None,  # Will be updated after Midtrans response
        )
        db.session.flush()  # This generates the ID but doesn't commit yet

```

```
        # Prepare Midtrans parameters
        params = {
            "payment_type": payment_method,
            "transaction_details": {
                "order_id": payment.reference_id,
                "gross_amount": payment.amount
            },
            "customer_details": {
                "first_name": konsultasi.pasien.name,
                "email": konsultasi.pasien.email,
            },
            "item_details": [{
                "id": f"CONS-{konsultasi_id}",
                "price": payment.amount,
                "quantity": 1,
                "name": f"Consultation with Dr. {konsultasi.psikolog.name}"
            }],
            "expiry": {
                "duration": 24,
                "unit": "hours"
            }
        }

        # Add specific payment details based on method
        if payment_method == "bank_transfer":
            params["bank_transfer"] = {"bank": payment_provider}
        elif payment_method == "echannel":
            params["echannel"] = {"bill_info1": "Consultation Payment"}
        elif payment_method == "gopay":
            params["gopay"] = {"enable_callback": True}

        # Create transaction in Midtrans
        midtrans_client = get_midtrans_client()
        charge_response = midtrans_client.charge(params)

        # Update payment with provider reference
        payment.provider_reference = charge_response.get('transaction_id')

        # Update VA number if available
        if payment_method in ['bank_transfer']:
            payment.va_number = charge_response.get('va_numbers', [{}])[0].get('va_number', None)

        db.session.commit()

        return jsonify({
            'status': 'success',
            'data': {
                'payment_id': payment.id,
                'reference_id': payment.reference_id,
                'amount': payment.amount,
                'payment_type': payment_method,
                'transaction_id': charge_response.get('transaction_id'),
                'payment_code': charge_response.get('payment_code'),
                'va_numbers': charge_response.get('va_numbers'),
                'actions': charge_response.get('actions'),
                'expiry_time': payment.payment_expired.isoformat()
            }
        }), 200

    except Exception as e:
        db.session.rollback()
        # Log the error for debugging
        print(f"Error creating payment: {str(e)}")
        return jsonify({
            'status': 'error',
            'message': str(e)
        }), 500
```

Endpoint Flask ini digunakan untuk membuat pembayaran konsultasi berdasarkan konsultasi_id dan hanya bisa diakses oleh pengguna yang login (JWT). Kode mengambil user_id dari token, data konsultasi dari database, lalu membaca metode dan penyedia pembayaran dari request. Setelah membuat entri pembayaran awal, sistem menyusun parameter dan mengirim transaksi ke Midtrans. Respons dari Midtrans (seperti transaction_id, va_number) digunakan untuk memperbarui data pembayaran sebelum disimpan ke database. Jika sukses, detail pembayaran dikembalikan, jika gagal, transaksi dibatalkan dan error dikirimkan. Proses ini memastikan setiap transaksi tercatat dengan referensi unik dan masa berlaku tertentu. Seluruh alur juga menangani skenario pembayaran berbeda seperti transfer bank atau e-wallet.

13. Route Chatbot

```python
@chatbot_bp.route('/chat', methods=['POST'])
def chat():

    try:
        # Get message from request
        data = request.get_json()
        if not data or 'message' not in data:
            return jsonify({'error': 'No message provided'}), 400

        message = data['message']

        # Load resources
        model, words, classes, intents = load_chatbot_resources()
        if None in (model, words, classes, intents):
            return jsonify({'error': 'Failed to load chatbot resources'}), 500

        # Get prediction
        ints = predict_class(message, model, words, classes)

        # Get response
        response = get_response(intents, ints)

        result = {
            'response': response,
            'intent': ints[0]['intent'] if ints else 'unknown',
            'confidence': float(ints[0]['probability']) if ints else 0.0
        }

        return jsonify(result), 200

    except Exception as e:
        error_msg = {'error': str(e)}
        return jsonify(error_msg), 500
```

Endpoint /chat dengan metode POST ini digunakan untuk memproses interaksi chatbot berbasis teks dari pengguna. Sistem menerima input berupa JSON dengan key message, lalu memvalidasi keberadaan data tersebut. Selanjutnya, chatbot memuat semua resource penting seperti model machine learning, daftar kata (words), kelas intent (classes), dan kumpulan intent (intents) menggunakan fungsi load_chatbot_resources(). Jika ada resource yang gagal dimuat, maka akan dikembalikan respons error 500 dan kesalahan dicatat ke dalam log. Setelah resource berhasil dimuat, pesan dari pengguna diproses melalui fungsi predict_class() untuk mengklasifikasikan intent berdasarkan model yang sudah dilatih. Dari intent yang diprediksi, fungsi get_response() digunakan untuk memilih respons paling relevan dari daftar intent. Hasil akhir yang dikembalikan berupa JSON berisi response (jawaban dari chatbot), intent (nama intent yang dikenali), dan confidence (nilai probabilitas keyakinan model terhadap intent tersebut). Jika terjadi exception selama proses, sistem akan mencatat kesalahan menggunakan logging.error() dan mengembalikan respons error 500 kepada klien. Endpoint ini menjadi bagian utama dalam menjalankan fungsionalitas chatbot interaktif.

14. Route Konseling

```python
@consultation_bp.route('/consultations/check-booking', methods=['GET'])
@jwt_required()
def check_user_booking():
    """Check if user has any active bookings"""
    try:
        current_user_id = get_jwt_identity()

        # Cek pesanan yang aktif (status menunggu atau dikonfirmasi)
        active_bookings = JadwalKonsultasi.query.filter(
            JadwalKonsultasi.pasien_id == current_user_id,
            JadwalKonsultasi.status.in_([
                JadwalKonsultasi.STATUS_MENUNGGU,
                JadwalKonsultasi.STATUS_DIKONFIRMASI
            ]),
            JadwalKonsultasi.tanggal_konsultasi >= datetime.now().date()
        ).order_by(
            JadwalKonsultasi.tanggal_konsultasi,
            JadwalKonsultasi.waktu_konsultasi
        ).all()

        # Format hasil query
        bookings = [{
            'id': booking.id,
            'psikolog': {
                'id': booking.psikolog.id,
                'name': booking.psikolog.name
            },
            'tanggal': booking.tanggal_konsultasi.strftime('%Y-%m-%d'),
            'waktu': booking.waktu_konsultasi.strftime('%H:%M'),
            'durasi': booking.durasi,
            'jenis_konsultasi': booking.jenis_konsultasi,
            'status': booking.status,
            'biaya_konsultasi': booking.biaya_konsultasi,
            'status_pembayaran': booking.status_pembayaran,
            'is_upcoming': (
                datetime.combine(booking.tanggal_konsultasi, booking.waktu_konsultasi) >
                datetime.now()
            )
        } for booking in active_bookings]

        return jsonify({
            'has_active_bookings': len(bookings) > 0,
            'total_active_bookings': len(bookings),
            'bookings': bookings
        }), 200

    except Exception as e:
        logging.error(f"Error checking user bookings: {str(e)}")
        return jsonify({
            'message': f'Error checking user bookings: {str(e)}'
        }), 500
```

Endpoint GET /consultations/check-booking digunakan untuk memeriksa apakah pengguna saat ini memiliki jadwal konsultasi yang masih aktif, yaitu dengan status menunggu atau dikonfirmasi, dan tanggal konsultasinya belum lewat dari hari ini. Endpoint ini hanya dapat diakses oleh pengguna yang telah terautentikasi melalui JWT. Sistem akan melakukan query terhadap tabel JadwalKonsultasi untuk mencari semua pemesanan aktif milik pengguna, mengurutkannya berdasarkan tanggal dan waktu konsultasi, lalu memformat hasilnya ke dalam daftar berisi informasi lengkap seperti ID konsultasi, nama psikolog, tanggal dan waktu konsultasi, durasi,

jenis layanan, biaya, status pembayaran, serta indikator apakah konsultasi tersebut akan datang (is_upcoming). Hasilnya akan dikembalikan dalam bentuk JSON yang mencakup status apakah pengguna memiliki booking aktif, total booking aktif, dan daftar detail setiap booking. Jika terjadi kesalahan selama pemrosesan, sistem akan mengembalikan pesan error beserta log yang sesuai.

15. Route Chat Konseling

```python
@chat_bp.route('/chats/consultation/<int:consultation_id>', methods=['GET'])
@jwt_required()
def get_consultation_chats(consultation_id):
    """
    Retrieve chat messages for a specific consultation
    """
    try:
        current_user_id = get_jwt_identity()

        # Check if consultation exists
        consultation = JadwalKonsultasi.query.get_or_404(consultation_id)

        # Verify user's participation in consultation
        is_patient = str(consultation.pasien_id) == current_user_id
        is_psychologist = str(consultation.psikolog.user_id) == current_user_id

        if not (is_patient or is_psychologist):
            return jsonify({'message': 'Unauthorized to view chats for this consultation'}), 403

        # Optional pagination
        page = request.args.get('page', 1, type=int)
        per_page = request.args.get('per_page', 50, type=int)
        per_page = min(per_page, 100)  # Limit max messages per page

        # Optional last message timestamp for fetching newer messages
        last_message_timestamp = request.args.get('last_message_timestamp')

        # Query chat messages with pagination
        chats_query = Chat.query.filter_by(konsultasi_id=consultation_id)

        # If last_message_timestamp is provided, fetch only messages after that timestamp
        if last_message_timestamp:
            last_timestamp = datetime.fromisoformat(last_message_timestamp)
            chats_query = chats_query.filter(Chat.dikirim_pada > last_timestamp)

        # Set unread messages as read if viewer is not the sender
        current_user_type = 'pasien' if is_patient else 'psikolog'
        unread_messages = chats_query.filter(
            Chat.is_read == False,
            Chat.sender_type != current_user_type  # Only mark the other person's messages as read
        ).all()

        for message in unread_messages:
            message.is_read = True
            message.read_at = datetime.utcnow()

        if unread_messages:
            db.session.commit()

        # Order and paginate
        chats_query = chats_query.order_by(Chat.dikirim_pada)
        paginated_chats = chats_query.paginate(page=page, per_page=per_page)

        # Get sender details
        patient = consultation.pasien
        psychologist = consultation.psikolog

        # Format chat messages
        chat_messages = [{
            'id': chat.id,
            'sender_id': chat.sender_id,
            'sender_type': chat.sender_type,
            'sender_name': (patient.name if chat.sender_type == 'pasien' else psychologist.name),
```

```
            'message': chat.pesan,
            'timestamp': chat.dikirim_pada.isoformat(),
            'is_read': chat.is_read,
            'read_at': chat.read_at.isoformat() if chat.read_at else None
        } for chat in paginated_chats.items]

        return jsonify({
            'consultation_details': {
                'patient_id': patient.id,
                'patient_name': patient.name,
                'psychologist_id': psychologist.id,
                'psychologist_name': psychologist.name,
                'consultation_date': consultation.tanggal_konsultasi.strftime('%Y-%m-%d'),
                'consultation_time': consultation.waktu_konsultasi.strftime('%H:%M')
            },
            'messages': chat_messages,
            'total_messages': paginated_chats.total,
            'total_pages': paginated_chats.pages,
            'current_page': page
        }), 200

    except Exception as e:
        logging.error(f"Error retrieving chat messages: {str(e)}")
        return jsonify({'message': f'Error retrieving chat messages: {str(e)}'}), 500
```

Endpoint GET /chats/consultation/<consultation_id> berfungsi untuk mengambil seluruh pesan chat yang berkaitan dengan satu sesi konsultasi tertentu berdasarkan consultation_id. Endpoint ini menggunakan metode HTTP GET, yang menandakan bahwa operasinya hanya untuk membaca data (bukan mengubah). Ini penting karena nantinya akan ada endpoint POST terpisah yang khusus digunakan untuk mengirim atau menyimpan pesan baru. Endpoint ini dilindungi dengan autentikasi JWT (@jwt_required) dan hanya bisa diakses oleh pengguna yang terlibat langsung dalam sesi tersebut, yaitu pasien atau psikolog. Jika pengguna tidak memiliki izin, maka akan dikembalikan status 403 (Unauthorized). Endpoint ini juga mendukung paginasi melalui parameter page dan per_page, serta filter opsional last_message_timestamp untuk mengambil hanya pesan yang dikirim setelah waktu tertentu.

Selain mengambil data, endpoint ini juga menangani penandaan pesan sebagai terbaca (read). Ketika pengguna yang mengakses bukan pengirim pesan, sistem akan mengubah status is_read menjadi True dan mencatat waktu baca (read_at). Pesan-pesan yang berhasil diambil akan disusun dalam format JSON yang berisi informasi lengkap seperti ID dan nama pengirim, isi pesan, waktu pengiriman, status baca, serta detail jadwal konsultasi seperti nama pasien, nama psikolog, tanggal, dan jam konsultasi. Informasi paginasi juga disertakan, termasuk jumlah total pesan dan halaman. Jika terjadi error, akan dicatat log detail dan dikembalikan respons HTTP 500. Dengan struktur dan logika ini, endpoint GET ini berfungsi sebagai bagian utama dari sistem chat real-time untuk menampilkan riwayat komunikasi antar pengguna dalam sesi konsultasi.

```python
@chat_bp.route('/chats/consultation/<int:consultation_id>', methods=['POST'])
@jwt_required()
def send_chat_message(consultation_id):
    """
    Send a chat message for a specific consultation
    """
    try:
        current_user_id = get_jwt_identity()
        data = request.get_json()

        # Validate required fields
        if not data or 'message' not in data:
            return jsonify({'message': 'Message content is required'}), 400

        # Check if consultation exists and user is a participant
        consultation = JadwalKonsultasi.query.get_or_404(consultation_id)

        # Verify user's participation in consultation
        is_patient = str(consultation.pasien_id) == current_user_id
        is_psychologist = str(consultation.psikolog.user_id) == current_user_id

        if not (is_patient or is_psychologist):
            return jsonify({'message': 'Unauthorized to send message for this consultation'}), 403

        # Check consultation status
        if consultation.status not in [consultation.STATUS_MENUNGGU, consultation.STATUS_DIKONFIRMASI]:
            return jsonify({'message': 'Cannot send messages for inactive consultations'}), 400

        # Determine sender type
        sender_type = 'pasien' if is_patient else 'psikolog'

        # Validate message length
        if len(data['message'].strip()) == 0:
            return jsonify({'message': 'Message cannot be empty'}), 400

        if len(data['message']) > 1000:  # Example max length
            return jsonify({'message': 'Message exceeds maximum length of 1000 characters'}), 400

        # Create new chat message
        chat_message = Chat(
            konsultasi_id=consultation_id,
            sender_id=current_user_id,
            sender_type=sender_type,
            pesan=data['message'].strip(),
            is_read=False  # New message is unread by default
        )

        db.session.add(chat_message)
        db.session.commit()

        return jsonify({
            'message': 'Chat message sent successfully',
            'chat_id': chat_message.id,
            'sender_type': sender_type
        }), 201

    except Exception as e:
        db.session.rollback()
        logging.error(f"Error sending chat message: {str(e)}")
        return jsonify({'message': f'Error sending chat message: {str(e)}'}), 500
```

Endpoint /chats/consultation/<int:consultation_id> dengan metode POST memungkinkan pengguna yang telah login (dengan JWT) untuk mengirim pesan chat dalam sesi konsultasi tertentu. Setelah mengambil user_id dari token dan memvalidasi keberadaan data pesan, sistem memastikan bahwa pengguna adalah pasien atau psikolog yang terlibat dalam konsultasi tersebut. Jika pengguna tidak terlibat atau status konsultasi tidak aktif (bukan menunggu atau dikonfirmasi), maka pengiriman pesan akan ditolak. Pesan juga divalidasi agar tidak kosong dan tidak melebihi 1000 karakter. Jika valid, sistem membuat objek Chat baru dengan status belum dibaca (is_read=False) dan menyimpannya ke database. Jika berhasil, sistem mengembalikan respons sukses beserta ID pesan dan tipe pengirim; jika gagal, terjadi rollback dan error dicatat untuk debugging.

16. Route Deteksi Basic Emotion

```python
@video_bp.route('/detect', methods=['POST'])
@jwt_required()
def detect_video():
    """Process uploaded video file for emotion detection and immediate attachment processing."""
    logger.debug("=== Starting Video Detection ===")

    try:
        if processor is None:
            logger.error("Video processor not initialized")
            return jsonify({'error': 'Video processor not initialized'}), 500

        current_user_id = get_jwt_identity()
        logger.debug(f"User ID: {current_user_id}")
        logger.debug(f"Content Type: {request.content_type}")
        logger.debug(f"Files: {request.files}")
        logger.debug(f"Form: {request.form}")

        # Validate request
        if not request.content_type or not request.content_type.startswith('multipart/form-data'):
            logger.error(f"Invalid content type: {request.content_type}")
            return jsonify({'error': 'Content-Type must be multipart/form-data'}), 400

        if 'video' not in request.files:
            logger.error("No video file in request")
            return jsonify({'error': 'No video file provided'}), 400

        if 'stimuli' not in request.form:
            logger.error("No stimuli in request")
            return jsonify({'error': 'No stimuli type provided'}), 400

        video_file = request.files['video']
        stimuli = request.form['stimuli']

        logger.debug(f"Video filename: {video_file.filename}")
        logger.debug(f"Stimuli: {stimuli}")

        if video_file.filename == '':
            logger.error("Empty video filename")
            return jsonify({'error': 'Empty video filename'}), 400

        if stimuli not in ATTACHMENT_MAPPING:
            logger.error(f"Invalid stimuli: {stimuli}")
            return jsonify({'error': 'Invalid stimuli type'}), 400

        # Create directories
        temp_dir = os.path.join('uploads', 'videos')
        storage_dir = os.path.join('storage', 'videos')
        os.makedirs(temp_dir, mode=0o755, exist_ok=True)
        os.makedirs(storage_dir, mode=0o755, exist_ok=True)

        # Process video
        timestamp = datetime.now(local_timezone).strftime('%Y%m%d_%H%M%S')
        temp_filename = f"temp_video_{current_user_id}_{timestamp}.mp4"
        storage_filename = f"video_{current_user_id}_{timestamp}.mp4"

        temp_path = os.path.join(temp_dir, temp_filename)
        storage_path = os.path.join(storage_dir, storage_filename)

        logger.debug(f"Saving temp file to: {temp_path}")
        video_file.save(temp_path)

        try:
            results = processor.process_video(temp_path)
            if not results:
                raise ValueError("No valid frames processed")

            # Calculate results
            emotion_counts = {}
            total_confidence = 0

            for result in results:
                emotion_label = result['emotion_label']
                emotion_counts[emotion_label] = emotion_counts.get(emotion_label, 0) + 1
                total_confidence += result['confidence']

            dominant_emotion = max(emotion_counts.items(), key=lambda x: x[1])[0]
            avg_confidence = total_confidence / len(results) if results else 0

            # Save to permanent storage
            shutil.copy2(temp_path, storage_path)
            relative_path = os.path.join('videos', storage_filename)

```

```
            # Save to database
            history = HistoryDetection(
                user_id=current_user_id,
                hasil=determine_attachment_style(stimuli, dominant_emotion),
                emotion=dominant_emotion,
                stimuli=stimuli,
                video_path=relative_path
            )
            db.session.add(history)
            db.session.commit()

            # Get the history ID after commit
            history_id = history.id

            # Always process attachment results
            logger.debug(f"Processing attachment for user {history_id} with dominant emotion {dominant_emotion}")

            # Prepare response with history_id
            response_data = {
                'status': 'success',
                'history_id': history_id,  # Add history ID to response
                'timestamp': timestamp,
                'video_path': relative_path,
                'summary': {
                    'total_frames': len(results),
                    'emotions': emotion_counts,
                    'dominant_emotion': dominant_emotion,
                    'confidence': avg_confidence,
                    'attachment_style': history.hasil
                }
            }

            logger.debug(f"Video processing completed successfully with history ID: {history_id}")
            return jsonify(response_data), 200

        except Exception as e:
            logger.error(f"Processing error: {str(e)}")
            raise

    except Exception as e:
        logger.error(f"Error in detect_video: {str(e)}")
        return jsonify({'error': str(e)}), 500

    finally:
        # Cleanup temp file
        if 'temp_path' in locals() and os.path.exists(temp_path):
            try:
                os.remove(temp_path)
                logger.debug("Temporary file cleaned up")
            except Exception as e:
                logger.warning(f"Failed to remove temp file: {str(e)}")
```

Fungsi endpoint POST /detect digunakan untuk memproses file video yang diunggah oleh pengguna sebagai bagian dari deteksi emosi berdasarkan stimuli tertentu, dan secara langsung menyimpulkan gaya keterikatan (attachment style) pengguna berdasarkan dominasi emosi dari ekspresi wajah. Hanya pengguna yang telah terautentikasi (dengan JWT) yang dapat mengakses endpoint ini. Proses diawali dengan validasi terhadap Content-Type dan keberadaan file video serta stimuli, lalu video disimpan sementara, dianalisis frame demi frame oleh modul processor, dan hasil emosi dominan dihitung bersama dengan rata-rata tingkat kepercayaannya. Setelah itu, hasil deteksi disimpan ke database (HistoryDetection) bersama informasi video dan gaya keterikatan yang disimpulkan. File video juga dipindahkan ke folder penyimpanan permanen, dan ID histori dikembalikan sebagai respons. Endpoint ini juga dilengkapi dengan sistem logging detail dan pembersihan otomatis terhadap file sementara untuk menjaga efisiensi dan keamanan sistem.

17. Route Riwayat Deteksi

```python
@attachment_bp.route('/attachment-result', methods=['GET'])
@jwt_required()
def get_attachment_results():
    """Get all attachment results for current user"""
    current_user_id = get_jwt_identity()

    try:
        results = AttachmentResult.query.filter_by(user_id=current_user_id).order_by(
            AttachmentResult.created_at.desc()
        ).all()

        response_data = [{
            'id': result.id,
            'final_result': result.final_result,
            'secure_count': result.secure_count,
            'insecure_anxious_count': result.insecure_anxious_count,
            'insecure_avoidant_count': result.insecure_avoidant_count,
            'created_at': result.created_at.isoformat()
        } for result in results]

        return jsonify(response_data), 200

    except Exception as e:
        logging.error(f"Error fetching attachment results for user ID {current_user_id}: {str(e)}")
        logging.debug("Traceback:\n" + traceback.format_exc())
        return jsonify({'message': f'Error fetching attachment results: {str(e)}'}), 500
```

Fungsi endpoint GET /attachment-result ini digunakan untuk mengambil seluruh hasil tes attachment milik pengguna yang sedang login. Endpoint ini dilindungi oleh @jwt_required(), yang berarti hanya pengguna terautentikasi melalui token JWT yang bisa mengaksesnya. Proses dimulai dengan mengambil user_id dari token JWT menggunakan get_jwt_identity(). Kemudian sistem melakukan query ke database untuk mengambil semua data dari tabel AttachmentResult yang berelasi dengan pengguna tersebut. Query ini menyertakan order_by(AttachmentResult.created_at.desc()) untuk mengurutkan hasil berdasarkan waktu pembuatan paling baru ke paling lama.

Setelah data diperoleh, hasilnya diformat menjadi list of dictionaries (response_data) berisi informasi penting dari masing-masing entri, termasuk id, hasil akhir (final_result), jumlah masing-masing tipe keterikatan (secure, insecure anxious, insecure avoidant), dan waktu pembuatan (created_at) dalam format ISO. Jika proses berhasil, sistem mengembalikan hasil tersebut dalam bentuk JSON dengan status HTTP 200. Namun jika terjadi kesalahan selama proses (seperti error koneksi database atau kesalahan tak terduga lainnya), maka error akan dicatat melalui logging.error, termasuk stack trace via logging.debug, dan sistem akan merespons dengan status HTTP 500 serta pesan error yang sesuai.

18. Route Jadwal Psikolog

```python
@jadwal_bp.route('/jadwal', methods=['GET'])
@jwt_required()
def get_jadwal_by_user_or_psikolog():
    """
    Get schedules (jadwal) based on user_id or psikolog_id
    User can query their own schedules or specific psychologist schedules
    Requires JWT authentication
    """
    try:
        # Get the current user's ID from JWT token
        current_user_id = get_jwt_identity()
        logging.info(f"User ID {current_user_id} requesting jadwal information")

        # Ensure user exists
        user = User.query.get(current_user_id)
        if not user:
            logging.warning(f"User ID {current_user_id} not found in database")
            return jsonify({'message': 'User not found'}), 404

        # Get query parameters
        user_id = request.args.get('user_id', type=int)
        psikolog_id = request.args.get('psikolog_id', type=int)

        # If neither parameter is provided, default to current user's ID
        if not user_id and not psikolog_id:
            user_id = int(current_user_id)
            logging.info(f"No parameters provided, defaulting to current user ID: {user_id}")

        # If user_id is provided, find all psikolog associated with this user
        if user_id:
            # Check if requested user_id matches the authenticated user or if user is a psychologist
            if int(user_id) != int(current_user_id) and user.role != 'psikolog':
                logging.warning(f"User {current_user_id} attempted to access schedules for user {user_id}")
                return jsonify({'message': 'You can only view your own schedules unless you are a psychologist'}), 403

            psikolog_list = Psikolog.query.filter_by(user_id=user_id).all()
            if not psikolog_list:
                return jsonify({'message': 'No psychologist profile found for this user'}), 404

            # Get jadwal for all psikolog profiles of this user
            result = []
            for psikolog in psikolog_list:
                jadwal_data = _get_psikolog_jadwal(psikolog.id)
                if jadwal_data:
                    result.append(jadwal_data)

            return jsonify({'jadwal': result}), 200

        # If psikolog_id is provided, get jadwal for just that specific psychologist
        elif psikolog_id:
            # Anyone can view a psychologist's schedule
            psikolog = Psikolog.query.get(psikolog_id)
            if not psikolog:
                return jsonify({'message': 'Psychologist not found'}), 404

            jadwal_data = _get_psikolog_jadwal(psikolog_id)
            if not jadwal_data:
                return jsonify({'message': 'No schedule found for this psychologist'}), 404

            return jsonify({'jadwal': jadwal_data}), 200

    except Exception as e:
        logging.error(f"Error retrieving jadwal: {str(e)}")
        return jsonify({'message': f'Error retrieving jadwal: {str(e)}'}), 500

def _get_psikolog_jadwal(psikolog_id):
    """Helper function to get jadwal for a specific psikolog"""
    psikolog = Psikolog.query.get(psikolog_id)
    if not psikolog:
        return None

    working_days = WorkingDay.query.filter_by(psikolog_id=psikolog_id).all()
    if not working_days:
        return None

    # Format the jadwal data
    days_data = []
    for day in working_days:
        hours = []
        for hour in day.working_hours:
            hours.append({
                'id': hour.id,
                'jam_mulai': hour.jam_mulai.strftime('%H:%M'),
                'jam_selesai': hour.jam_selesai.strftime('%H:%M')
            })

        days_data.append({
            'id': day.id,
            'hari': day.hari,
            'hours': hours
        })

    return {
        'psikolog_id': psikolog_id,
        'psikolog_name': psikolog.name,
        'working_days': days_data
    }
```

Kode sebelumnya mendefinisikan endpoint API GET /jadwal yang memungkinkan pengguna, setelah melalui autentikasi JWT, untuk mengambil data jadwal praktik psikolog berdasarkan user_id atau psikolog_id yang dikirim melalui parameter query. Jika parameter tidak diberikan, maka sistem akan default menggunakan ID pengguna yang sedang login. Jika user_id diberikan, sistem akan memastikan bahwa pengguna yang meminta data adalah dirinya sendiri atau memiliki peran sebagai psikolog, kemudian mengambil seluruh profil psikolog terkait dan mengembalikan jadwal praktiknya. Jika psikolog_id diberikan, sistem langsung mengambil jadwal praktik dari psikolog tersebut. Fungsi bantu _get_psikolog_jadwal digunakan untuk membentuk struktur data jadwal, mencakup hari kerja dan jam praktik. Validasi dilakukan untuk memastikan keberadaan pengguna dan psikolog sebelum data dikembalikan, dan semua kesalahan yang muncul akan di-log serta diberikan respons error yang sesuai.

19. Route Riwayat Pembayaran

```python
@payment_attachment_bp.route('/attachment-history', methods=['GET'])
@jwt_required()
def get_attachment_payment_history():
    """
    Get attachment payment transaction history for current user
    """
    try:
        # Get current user ID from JWT
        current_user_id = get_jwt_identity()

        # Get optional query parameters for filtering
        status = request.args.get('status')
        start_date = request.args.get('start_date')
        end_date = request.args.get('end_date')

        # Base query for current user
        query = PaymentAttachment.query.filter(
            PaymentAttachment.user_id == current_user_id
        )

        # Apply filters if provided
        if status:
            query = query.filter(PaymentAttachment.status == status)

        if start_date:
            try:
                start_date_obj = datetime.strptime(start_date, '%Y-%m-%d')
                query = query.filter(PaymentAttachment.payment_created >= start_date_obj)
            except ValueError:
                return jsonify({
                    'status': 'error',
                    'message': 'Invalid start_date format. Use YYYY-MM-DD'
                }), 400

        if end_date:
            try:
                end_date_obj = datetime.strptime(end_date, '%Y-%m-%d')
                end_date_obj = datetime.combine(end_date_obj.date(), datetime.max.time())
                query = query.filter(PaymentAttachment.payment_created <= end_date_obj)
            except ValueError:
                return jsonify({
                    'status': 'error',
                    'message': 'Invalid end_date format. Use YYYY-MM-DD'
                }), 400

        # Order by creation date (newest first)
        query = query.order_by(desc(PaymentAttachment.payment_created))

        # Execute query
        transactions = query.all()

        # Format response data
        response_data = {
            'status': 'success',
            'data': []
        }

        for payment in transactions:
            # Get attachment result data
            attachment_result = AttachmentResult.query.get(payment.attachment_result_id)
```

```
            payment_data = {
                'id': payment.id,
                'reference_id': payment.reference_id,
                'amount': payment.amount,
                'payment_method': payment.payment_method,
                'payment_provider': payment.payment_provider,
                'status': payment.status,
                'description': payment.description,
                'created_at': payment.payment_created.isoformat(),
                'expired_at': payment.payment_expired.isoformat() if payment.payment_expired else None,
                'paid_at': payment.payment_paid.isoformat() if payment.payment_paid else None,
            }

            # Add attachment result data if available
            if attachment_result:
                payment_data['attachment_result'] = {
                    'id': attachment_result.id,
                    'final_result': attachment_result.final_result,
                    'secure_count': attachment_result.secure_count,
                    'insecure_anxious_count': attachment_result.insecure_anxious_count,
                    'insecure_avoidant_count': attachment_result.insecure_avoidant_count,
                    'created_at': attachment_result.created_at.isoformat() if attachment_result.created_at else None
                }

            response_data['data'].append(payment_data)

        return jsonify(response_data), 200

    except Exception as e:
        import logging
        logging.error(f"Error retrieving attachment payment history: {str(e)}")
        return jsonify({
            'status': 'error',
            'message': str(e)
        }), 500
```

Endpoint GET /attachment-history digunakan untuk mengambil riwayat transaksi pembayaran hasil attachment milik pengguna yang sedang login, dan hanya dapat diakses setelah autentikasi JWT berhasil melalui @jwt_required(). Setelah mendapatkan user_id dari token, sistem membentuk query ke tabel PaymentAttachment untuk mengambil data berdasarkan pengguna, dengan opsi penyaringan tambahan menggunakan parameter status, start_date, dan end_date. Jika tanggal diberikan, sistem mengonversinya ke format tanggal dan memfilter berdasarkan waktu pembuatan transaksi (payment_created). Data kemudian diurutkan dari yang terbaru dan diformat ke dalam JSON, mencakup informasi pembayaran serta hasil attachment yang terkait melalui relasi ke AttachmentResult. Jika berhasil, sistem mengembalikan data dengan status 200 OK, dan jika terjadi kesalahan seperti parsing tanggal atau kegagalan query, sistem mencatat log error dan mengirimkan respons error 500.

Lampiran 5. 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	:	EC002025075489, 25 Juni 2025
Pencipta		
Nama	:	**Firda Aulia Rakhmah, Dyah Apriliani, S.T., M.Kom. dkk**
Alamat	:	Jalan H.Umar Said, RT.03/RW.03, No.08, Desa Kertasinduyasa, Jatibarang, Kab. Brebes, Jawa Tengah, 52261
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 Pendukung Kesehatan Mental Dengan AI Real-Time Facial Emotion Recognition Menggunakan Arsitektur Long Short-Term Memory (LSTM)**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	25 Juni 2025, di Kota Tegal
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor Pencatatan	:	000915750

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 6. Lembar Bimbingan

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama : **Firda Aulia Rakhmah**
Nim : **21090119**
No. Ponsel : **085540407654**
Judul : **Aplikasi Pendukung Kesehatan Mental Dengan *AI Real-Time Facial Emotion Recognition* Menggunakan Arsitektur *Long Short-Term Memory* (LSTM)**
Dosen Pembimbing I : **Dyah Apriliani, S.T., M.Kom.**

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1.	20 Maret 2025	Aplikasi	- Fokuskan fitur utama dan model deteksi - Validasi role aplikasi	
2.	16 April 2025	Aplikasi	Penyempurnaan role user	
3.	21 Mei 2025	Aplikasi	Penyempurnaan role psikolog	
4.	4 Juni 2025	Aplikasi	Penambahan fitur Riwayat dan persiapan HKI.	
5.	20 Juni 2025	Aplikasi, HKI	Manual Book dan Dokumen Teknis	
6.	23 Juni 2025	Laporan Bab 1	-	
7.	25 Juni 2025	Laporan Bab 2, Bab 3, Lampiran	Bab 2	
8.	26 Juni 2025	Laporan Bab 2	Acc Laporan	

Tegal, 26 Juni 2025
Dosen Pembimbing I,

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

SARJANA TERAPAN TEKNIK INFORMATIKA
POLITEKNIK HARAPAN BERSAMA

LEMBAR BIMBINGAN SKRIPSI

Nama : **Firda Aulia Rakhmah**
Nim : **21090119**
No. Ponsel : **085540407654**
Judul : **Aplikasi Pendukung Kesehatan Mental Dengan *AI Real-Time Facial Emotion Recognition* Menggunakan Arsitektur *Long Short-Term Memory* (LSTM)**
Dosen Pembimbing II : **Hepatika Zidny Ilmadina, S.Pd., M.Kom.**

No	Tanggal	Pemeriksaan	Perbaikan Yang Perlu Dilakukan	Paraf Pembimbing
1.	21 Maret 2025	Aplikasi	Melengkapi fitur.	
2.	21 April 2025	Aplikasi	Penyempurnaan aplikasi.	
3.	19 Mei 2025	Aplikasi	Penyempurnaan aplikasi.	
4.	2 Juni 2025	Aplikasi	Penyempurnaan aplikasi.	
5.	17 Juni 2025	HKI	Manual Book dan Dokumen Teknis	
6.	20 Juni 2025	Laporan Bab 1	-	
7.	15 Juni 2025	Laporan Bab 2	Pengujian Aplikasi	
8.	26 Juni 2025	Laporan Bab 3	Acc Laporan	

Tegal, 26 Juni 2025

Dosen Pembimbing II,

Hepatika Zidny Ilmadina, S.Pd., M.Kom.
NIPY. 09.015.225