

CIE 511 - PEMROGRAMAN MOBILE

Laporan Akhir Project Pemrograman Mobile

Aplikasi Mobile Diagnosis Kesehatan Mental Anak Remaja

Disusun Oleh :

NIM

20210801300

20210801368

20210801330

20210801360

20210801364

Nama Lengkap

Ricky Salim

Fikri Alam Arya Putra

Baby Anjeli

Adillah Saudah Shofa

Sulistiyawati

Dosen Pengampu :

7174 - Ir. Sawali Wahyu, S.Kom., M.Kom

Daftar Isi

BAB I	4
PENDAHULUAN	4
1. Abstrak	4
2. Latar Belakang dan Tujuan	4
2.1. Latar Belakang	4
2.2. Rumusan Masalah	5
2.3. Tujuan dan Manfaat	5
2.4. Batasan Masalah	6
3. Penjelasan Tambahan Spesifikasi Kebutuhan	6
A) Spesifikasi Fitur Tambahan	7
B) Spesifikasi Bonus yang Dikerjakan	7
4. Alur Pembuatan Program sesuai Metode Pengembangan Perangkat Lunak	8
4.1. Gejala Kesehatan Mental	8
4.2. Kategori Gangguan	8
4.3. Forward Chaining	9
4.4. Certainty Factor	9
4.5. Perhitungan Certaint Factor	10
5. Class Diagram	12
 BAB II.....	13
LANDASAN TEORI.....	13
2.1 Teori Teori Khusus	13
2.2 Teori Teori Umum	14
 BAB III	17
STRUKTUR MODUL DAN LOGIKA.....	17
3.1 Struktur Logika Program	17
A. Output Modul 1 Splash Screen	17
B. Output Modul 2 Login & Register	18
C. Output Modul Main Menu	19

D. Test Modul	20
a. Test Modul 1 Splash Screen.....	20
b. Test Modul 2 Login & Register	22
c. Test Modul 3 Main Menu	22
E. Pengujian Skenario	37
 BAB IV	42
HASIL DAN PEMBAHASAN.....	42
4. Pembagian Kerja dalam Kelompok	42
A. Lampiran	43
Notulen Rapat	43
Log Activity Anggota Kelompok	44
B. Cara Compile & Eksekusi Program	48
C. Dokumentasi Koordinasi.....	48
D. Penyertaan Model Analisis	49
E. Requirement Systems	49
F. Desain Perancangan Sistem.....	51
G. Desain UI / UX	56
H. Arsitektur Aplikasi.....	58
I. Flow Tampilan Akhir Program.....	59
J. Programming Source Code dan Database Design.....	64
 BAB V	242
KESIMPULAN DAN SARAN.....	242
5. Kesimpulan Dan Saran	242
a. Kesimpulan	242
b. Saran	242
 DAFTAR PUSTAKA	243

BAB 1

PENDAHULUAN

1. Abstrak

Masalah kesehatan mental pada anak, seperti stres, kecemasan, atau depresi adalah sesuatu yang nyata dan masalah tersebut sama pentingnya dengan masalah kesehatan fisiknya. Namun, banyak anak yang tidak mendapatkan perawatan yang seharusnya. Sehingga dibutuhkan sebuah teknologi kecerdasan buatan seperti sistem pakar yang merupakan bagian dari ilmu komputer yang mengadopsi pengetahuan manusia dan membuat agar mesin (komputer) dapat menyelesaikan masalah seperti sebaik yang dilakukan oleh para ahli. Penelitian ini mengembangkan sistem pakar untuk mendiagnosis gangguan mental anak menggunakan metode Forward Chaining dan Certainty Factor berbasis android, dimana setiap data gejala memiliki nilai belief sebagai nilai awal untuk mendapatkan hasil dalam metode Forward Chaining dan Certainty Factor.

Kata kunci - Sistem Pakar, Gangguan Mental, Forward Chaining, Certainty Factor, Android

2. Latar Belakang dan Tujuan

2.1 Latar Belakang

Mental Illness atau penyakit mental merupakan penyakit kejiwaan yang melibatkan perubahan emosi, pemikiran, perilaku ataupun gabungan dari ketiganya. Mental Illness sering dikaitkan dengan stress atau depresi. Gangguan kejiwaan dapat menyerang semua usia terutama remaja pada kisaran usia 18-21 tahun yang berada pada masa krusialnya.

Setiap orang mengalami kesehatan mental dari yang baik sampai yang buruk. Depresi sebesar 6.2% dimiliki oleh remaja pada usia 15-24 tahun. Kecenderungan untuk menyakiti diri sendiri (self harm) hingga bunuh diri akan dialami saat depresi berat. Kasus bunuh diri sebesar 80-80% akibat dari depresi dan kecemasan. Bunuh diri yang disebabkan oleh depresi dan stres sebanyak 16% dari remaja berusia 10-19 tahun.

Berdasarkan latar belakang tersebut, maka dirancang suatu sistem pakar yang dapat digunakan untuk membantu pasien dalam memperoleh informasi mengenai penyakit yang

diderita dengan melakukan konsultasi di dalam sistem tersebut. Sistem pakar ini dirancang untuk mengadopsi kemampuan seorang pakar yaitu dokter atau tenaga medis. Dengan menerapkan metode Forward chaining yang menggunakan teknik pencarian atau teknik pelacakan maju berdasarkan informasi yang ada dan digabungkan dengan rule/aturan untuk menghasilkan kesimpulan dan tujuan. Dengan metode Forward chaining ini komputer akan menganalisa permasalahan dengan mencari fakta yang cocok pada bagian IF dari aturan IF-THEN. Metode forward chaining juga mampu menyediakan berbagai informasi dengan hanya jumlah kecil dari data. Metode forward chaining juga mampu menyediakan berbagai informasi dengan hanya jumlah kecil dari data. Sistem yang dirancang ini berbasis Android dan memanfaatkan bahasa pemrograman Android Studio kemudian disimpan kedalam database beserta tabel-tabel untuk mendukung dalam penampungan data.

2.2 Rumusan Masalah

Berdasarkan permasalahan yang telah dijelaskan pada latar belakang diatas, maka rumusan masalah dari laporan ini adalah :

1. Bagaimana rancangan dari Pengembangan Aplikasi Diagnosis Kesehatan Mental pada Anak Remaja Menggunakan Metode Forward Chaining berbasis android ?
2. Bagaimana penerapan metode Forward Chaining dalam suatu aplikasi sistem pakar untuk mendiagnosa Kesehatan Mental Anak Remaja ?

2.3 Tujuan dan Manfaat

a) Tujuan

Yaitu mendiagnosis gangguan Kesehatan mental Anak Remaja menggunakan system pakar dengan metode Forward Chaining dan Certainty Factor

b) Manfaat

- Dapat digunakan sebagai pendiagnosis pada diri sendiri , keluarga atau orang terdekat apabila ada keanehan pada pola pikir, emosi dan perilaku. Sehingga pengguna dapat melakukan tindakan.
- Menyediakan solusi cepat dan mudah untuk mendeteksi tanda-tanda kesehatan mental yang memerlukan perhatian lebih lanjut.
- Membantu untuk memudahkan user dalam melakukan diagnosis awal pada penyakit Kesehatan Mental.

2.4 Batasan Masalah

- a) Aturan diagnostik yang digunakan dalam aplikasi bersifat umum dan mungkin tidak mencakup semua kasus atau variasi gejala yang mungkin terjadi pada anak remaja.
- b) Aplikasi tidak dirancang untuk menangani keadaan darurat atau krisis. Pengguna akan diingatkan untuk mencari bantuan langsung dari profesional atau lembaga kesehatan darurat jika diperlukan.
- c) Metode yang digunakan adalah metode Forward Chaining dan Certainly Factor dengan menghasilkan Keputusan

3. Spesifikasi Program

Dalam rangka pemenuhan tujuan-tujuan yang sebelumnya telah dijabarkan. Dibutuhkan daftar-daftar kriteria atau spesifikasi yang dibutuhkan sistem untuk memenuhi segala kebutuhan sistem dalam penggunaannya. Sehingga dalam peng-implementasiannya, penulis telah melakukan analisa dan identifikasi terkait dengan aspek-aspek tersebut yang mana akan dijabarkan berdasarkan menjadi spesifikasi diantaranya:

- a) Antarmuka Pengguna yang Ramah Pengguna: Antarmuka yang intuitif dan mudah digunakan, dengan navigasi yang jelas, agar remaja bisa menggunakannya tanpa kesulitan.
- b) Kuesioner Penilaian: Modul berisi serangkaian pertanyaan yang dirancang untuk mengidentifikasi gejala umum gangguan kesehatan mental pada remaja, seperti depresi, kecemasan, atau gangguan makan.
- c) Modul Edukasi: Informasi tentang berbagai aspek kesehatan mental, termasuk cara mengenali gejala, pentingnya mendapatkan bantuan, dan strategi pengelolaan.
- d) Sistem Rekomendasi: Berdasarkan jawaban dari kuesioner, aplikasi dapat memberikan rekomendasi tindakan awal, seperti menghubungi profesional kesehatan mental.
- e) Feedback dan Evaluasi: Fitur untuk memberikan umpan balik tentang aplikasi, yang bisa digunakan untuk perbaikan dan peningkatan berkelanjutan.

3.1 Spesifikasi Fitur Tambahan

Berikut merupakan beberapa spesifikasi fitur yang akan diimplementasikan pada program:

- a) Login, Pada tahap ini dilakukan penginputan informasi akun yang terdiri dari username dan password untuk menggunakan aplikasi. Antara username dan password keduanya saat digunakan untuk login harus sesuai dengan informasi akun yang dibuat pada saat Register.
- b) Register, Tahap register dilakukan agar informasi user dapat disimpan kedalam database sehingga user dapat menyimpan data hasil diagnosis dan melakukan login kembali. Informasi yang harus dimasukkan user diantaranya, Username, Password, dan Email.
- c) Welcome Page, Setelah melakukan Log-In, sistem akan menampilkan sarana edukasi mengenai Mental Illness berupa Video Learning , Relaxing Music dll

3.2 Spesifikasi Bonus Yang Dikerjakan

Dengan menggunakan aplikasi ini memiliki berbagai macam kelebihan dalam penggunaannya yaitu :

- a) Side Menu, Spesifikasi ini berfungsi untuk menyajikan pilihan menu untuk user ketika menjalankan aplikasi diagnosis. Dimana pada bagian ini terdapat tombol-tombol untuk menjalankan berbagai macam perintah pada program. Tombol yang ada pada side menu diantaranya, Vidio Learning, Relaxing Music, Diagnose, Breathing Exercise, and Help.
- b) Vidio Learning, merupakan salah satu fitur yang dibuat untuk pengenalan dan edukasi tentang mental illness pada remaja , dalam menu terdapat beberapa video yang telah disediakan.
- c) Relaxing Music , merupakan salah satu fitur yang dibuat untuk Merelexsasikan User/Pengguna , dimana dengan mendengarkan music dapat membuat hati dan pikiran menjadi tenang.
- d) Breathing Exercise, merupakan salah satu fitur yang dibuat seperti timer waktu, dimana pengguna dapat melakukan control nafas atau menahan nafas agar emosi lebih ke-contorl.
- e) Help, ini berfungsi untuk menyajikan pilihan menu untuk user diantaranya; Diagnose History, Profile, Sign Out dan Consultation

4. Alur Pembuatan Program sesuai Metode Pengembangan Perangkat Lunak

4.1 Gejala Kesehatan Mental

Pada pembuatan aplikasi mobile kesehatan mental pada remaja menggunakan Metode Forward Chaining dan Certainty Factor menggunakan gejala yang telah ditentukan oleh pakar. Adapun gejala dan kode gejala yang telah diberikan oleh pakar yaitu :

Tabel 1. Gejala

No	Kode	Gejala
1	G01	Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah.
2	G02	Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas.
3	G03	Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih.
4	G04	Pola makan yang tidak sehat, obsesi dengan berat badan.
5	G05	Kesulitan tidur atau tidur berlebihan.
6	G06	Kelelahan berkelanjutan, kurangnya motivasi.
7	G07	Perasaan tidak berharga atau bersalah tanpa alasan yang jelas.
8	G08	Kesulitan untuk fokus, mudah terdistraksi.
9	G09	Menghindari interaksi sosial, menyendiri.
10	G10	Pikiran untuk menyakiti diri sendiri, perilaku merusak diri.
11	G11	Penurunan atau kenaikan berat badan yang signifikan tanpa alasan yang jelas.
12	G12	Reaksi emosional yang berlebihan atau tidak sesuai.
13	G13	Tidak dapat melihat hal yang positif dari suatu kegiatan kejadian
14	G14	Merasa putus asa dan hilang harapan
15	G15	Merasa tidak ada hal yang dapat diharapkan di masa depan
16	G16	Merasa sulit untuk meningkatkan inisiatif dalam melakukan sesuatu

4.2 Kategori Gangguan

Pada pembuatan aplikasi mobile kesehatan mental pada remaja menggunakan Metode Forward Chaining dan Certainty Factor menggunakan kategori gangguan yang telah ditentukan oleh pakar. Adapun kategori gangguan dan kode gangguan yang telah diberikan oleh pakar yaitu :

Tabel 2. Gangguan

No	Kode	Gangguan	Point
1	KM01	Normal	0-20%
2	KM02	Depresi Ringan	>20-40%
3	KM03	Depresi Sedang	>40-60%
4	KM04	Depresi Berat Tanpa Gangguan Fungsional	>60-80%
5	KM05	Depresi Berat Dengan Gangguan Fungsional	>80%

4.3 Forward Chaining

Pada pembuatan aplikasi mobile kesehatan mental pada remaja menggunakan Metode Forward Chaining dan Certainty Factor pada tahap awal menentukan aturan forward Chainingng. Aturan forward chaining ditentukan berdasarkan rekomendasi yang telah diuji coba oleh pakar terhadap responden. Adapun aturan yang telah direkomendasikan yaitu :

Tabel 3. Aturan Forward Chaining

Gangguan	Aturan (Rules)
KM01	IF presentase $\leq 20\%$ THEN Diagnosis = Normal (Tidak ada Gangguan)
KM02	IF presentase $> 20\%$ dan $\leq 40\%$ THEN Diagnosis = Depresi Ringan
KM03	IF presentase $> 40\%$ dan $\leq 60\%$ THEN Diagnosis = Depresi Sedang
KM04	IF presentase $> 60\%$ dan $\leq 80\%$ THEN Diagnosis = Depresi Berat Tanpa Gangguan Fungsional
KM05	IF presentase $> 80\%$ THEN Diagnosis = Depresi Berat dengan Gangguan Fungsional

4.4 Certainty Factor

Pada pembuatan aplikasi mobile kesehatan mental pada remaja menggunakan Metode Forward Chaining dan Certainty Factor memiliki. Untuk nilai Certainty Factor gejala yang telah ditentukan pakar sebagai berikut :

Tabel 4. Certainty Faktor Gejala dari Pakar

No	Kode	Gejala	CF
1	G01	Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah.	0.7
2	G02	Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas.	0.8
3	G03	Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih.	0.6
4	G04	Pola makan yang tidak sehat, obsesi dengan berat badan.	0.5
5	G05	Kesulitan tidur atau tidur berlebihan.	0.6
6	G06	Kelelahan berkelanjutan, kurangnya motivasi.	0.7
7	G07	Perasaan tidak berharga atau bersalah tanpa alasan yang jelas.	0.75
8	G08	Kesulitan untuk fokus, mudah terdistraksi.	0.5
9	G09	Menghindari interaksi sosial, menyendiri.	0.65
10	G10	Pikiran untuk menyakiti diri sendiri, perilaku merusak diri.	0.85
11	G11	Penurunan atau kenaikan berat badan yang signifikan tanpa alasan yang jelas.	0.5
12	G12	Reaksi emosional yang berlebihan atau tidak sesuai.	0.55
13	G13	Tidak dapat melihat hal yang positif dari suatu kejadian	0.6

14	G14	Merasa putus asa dan hilang harapan	0.8
15	G15	Merasa tidak ada hal yang dapat diharapkan di masa depan	0.8
16	G16	Merasa sulit untuk meningkatkan inisiatif dalam melakukan sesuatu	0.6

Selain menentukan nilai Certainty factor pada gejala, pakar juga menentukan nilai Certainty Factor untuk memasukan user. User mengisikan nilai setiap gejala untuk menentukan hasil akhir gangguan. Adapun nilai masukkan Certainty Factor user sebagai berikut :

Tabel 5. Certainty Faktor Masukkan dari User

No	CF	Interpretasi
1	0	Tidak Pernah
2	0.03	Kadang-Kadang
3	0.3	Lumayan Sering
4	0.5	Sering sekali

4.5 Perhitungan Certainty Factor

Perhitungan CF kombinasi dilakukan dengan mengalikan CF pakar dengan CF user untuk setiap gejala, dan kemudian menggabungkan semua nilai CF kombinasi untuk mendapatkan CF total.

➤ **Gejala dan CF Masing-masing:**

G02: Perasaan sedih yang berkepanjangan (CF = 0.8)

G05: Kesulitan tidur atau tidur berlebihan (CF = 0.6)

G07: Perasaan tidak berharga atau bersalah tanpa alasan yang jelas (CF = 0.75)

a) **Langkah 1: Menghitung CF Kombinasi untuk G02 dan G05**

Rumus: CF kombinasi (CF1 dan CF2) = CF1 + CF2 * (1 - CF1)

Penerapan:

CF1 = 0.8 (untuk G02)

CF2 = 0.6 (untuk G05)

CF kombinasi = $0.8 + 0.6 * (1 - 0.8)$

CF kombinasi = $0.8 + 0.6 * 0.2$

CF kombinasi = $0.8 + 0.12$

CF kombinasi = 0.92

b) Langkah 2: Menghitung CF Kombinasi dengan G07

Rumus: CF kombinasi (CF sebelumnya dan CF3) = CF sebelumnya + CF3 * (1 - CF sebelumnya)

Penerapan:

CF sebelumnya = 0.92 (dari langkah 1)

CF3 = 0.75 (untuk G07)

CF kombinasi = $0.92 + 0.75 * (1 - 0.92)$

CF kombinasi = $0.92 + 0.75 * 0.08$

CF kombinasi = $0.92 + 0.06$

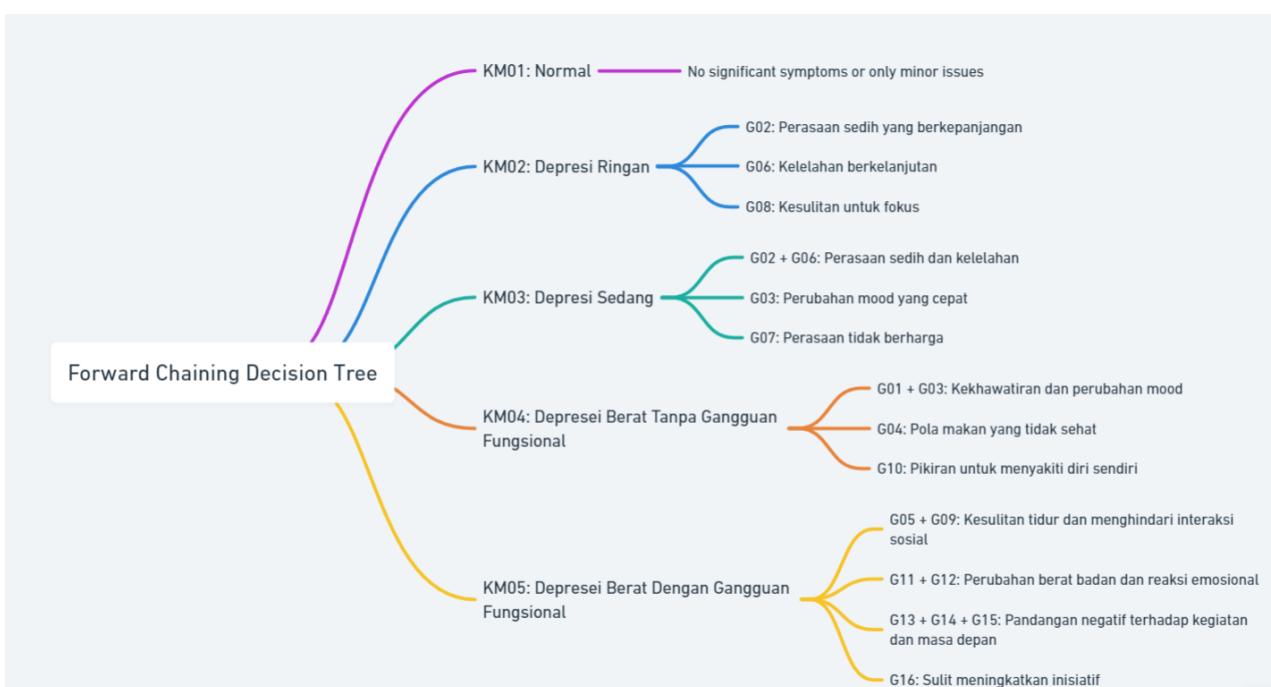
CF kombinasi = 0.98

c) Hasil:

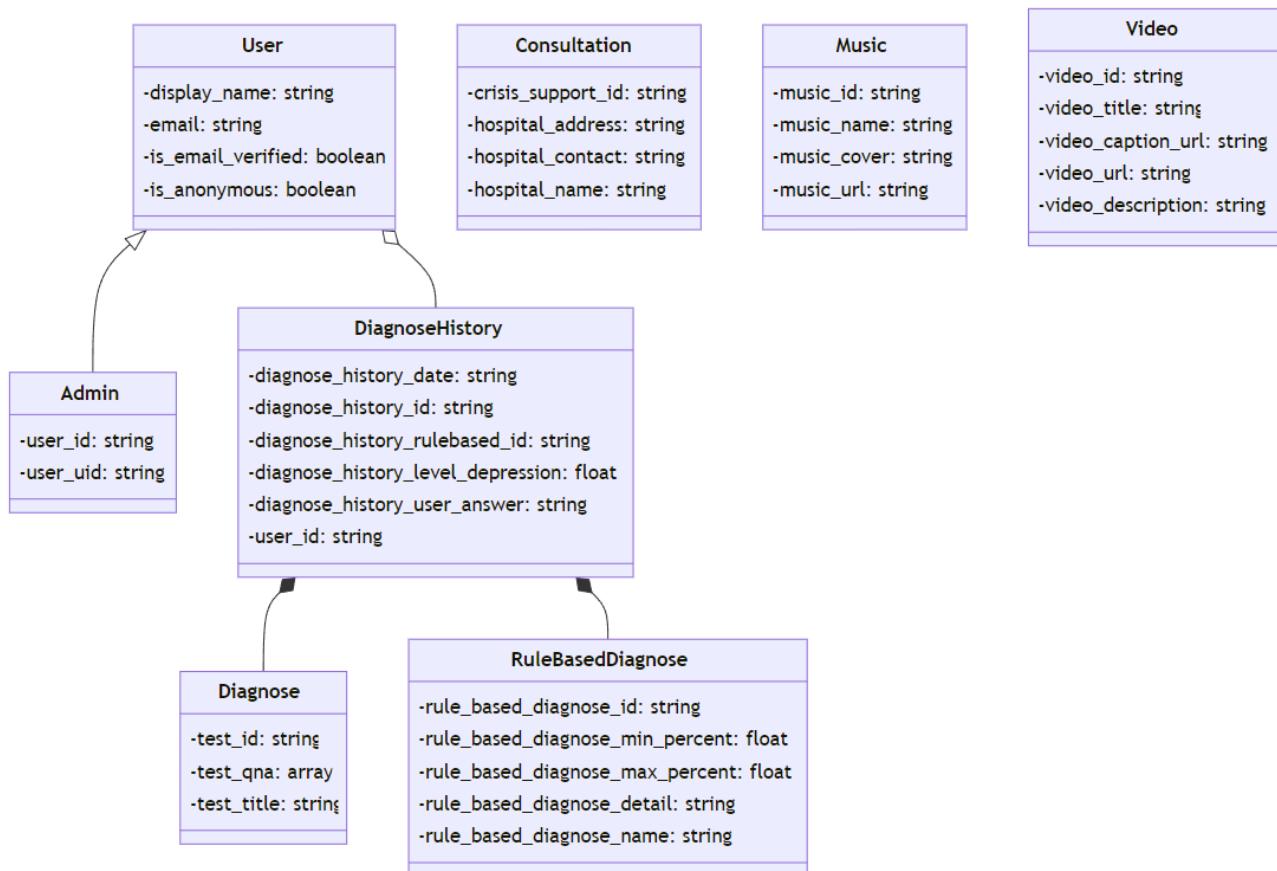
CF kombinasi untuk gejala G02, G05, dan G07 adalah 0.98. Dengan CF 0.98, yang berarti 98%, kita dapat menentukan bahwa pasien berada dalam kategori:

KM05: Depresi Berat Dengan Gangguan Fungsional (>80%)

Forward Chaining Decision Tree for Mental Health Symptoms



5. Class Diagram



BAB II

LANDASAN TEORI

2.1 Teori Teori Khusus

No	Peneliti	Judul Penelitian	Tahun	Hasil Penelitian
1.	Daniel Dwi Kurnia, Septi Andryana dan Aris Gunaryati	Sistem Pakar Untuk mendiagnosa Gangguan Kesehatan Mental Menggunakan Algoritma Genetika	2021	Sistem pakar utama ini dengan memanfaatkan teknik elemen deterministik untuk menganalisis perilaku disfungsional dapat membantu seseorang menganalisis jenis ketidakstabilan psikologis dengan efek samping yang jelas tanpa penilaian klinis, dan memberikan jawaban atas temuan tersebut.
2.	Winda Widya Ariestya,Yulia Eka Praptiningsih dan Muhammad Kasfi	Sistem Pakar Diagnosa Kesehatan Mental	2021	Sistem pakar spesialis untuk mendiagnosis masalah kesehatan psikologis untuk masalah mental alami, masalah mental gila, masalah mental masokis, masalah psikotik yang diringkas, membebani, masalah hipokondria disosiatif, kekacauan somatoform, kondisi perilaku dibuat menggunakan strategi pembubuhan ke depan. Pengujian yang telah dilakukan agar semua kapabilitas dapat berjalan sesuai dengan rencana yang telah dibuat.
3.	Dewi Kartika	Sistem Pakar Penyakit Lambung Dengan Metode <i>Forward chaining</i>	2017	1. Merencanakan kerangka kerja yang dilengkapi untuk mendiagnosis penyakit lambung oleh spesialis dan panduan tentang tindakan pencegahan. 2. Pemanfaatan strategi penjangkaran ke depan dalam mencapai tujuan atau mengejar pilihan dari kerangka induk yang direncanakan. 3. membantu spesialis dalam mendiagnosis penyakit dan pengobatannya. 4. Membantu masyarakat pada umumnya untuk mengetahui penyakit maag yang diderita secara efektif dan dini.

4.	Gideon Abram Filando Suwarso, Gregorius Satia Budhi2 dan Lily Puspa Dewi	Sistem Pakar untuk Penyakit Anak Menggunakan Metode <i>Forward chaining</i>	2015	1. Dengan adanya aplikasi ini, user bisa mengetahui penyakit apa yang sedang diderita oleh pasien. Karena proses diagnose secara garis besar bisa dilakukan secara <i>online</i> tanpa harus menemui dokter saat itu juga. Tetapi pasien tetap harus mendapatkan perawatan medis. 2. Berdasarkan hasil kuesioner yang disebar bisa disimpulkan bahwa aplikasi ini memiliki desain yang cukup bagus dan mudah digunakan oleh pengguna. 3. Berdasarkan hasil kuesioner dari segi kebutuhan, program ini cukup mampu memenuhi kebutuhan meskipun jenis penyakit harus dikembangkan kembali.
5.	Jesreel Surbakti dan Aqwam Rosadi Kardian	Sistem Pakar Kejiwaan dengan <i>Forward chaining</i> Berbasis Web	2017	1. Pembuatan algoritma <i>forward chaining</i> sudah sesuai dengan rancangan. 2. Penerapan siklus <i>forward chaining</i> ke dalam website sudah sesuai dengan rancangan. 3. Pengimplementasian sistem pakar pada sebuah website dengan membuat menu diagnosis yang memungkinkan pengguna untuk memilih gejala. Gejala yang dipilih oleh pengguna akan diproses dengan <i>forward chaining</i> sehingga sistem pakar

2.2 Teori Teori Umum

2.2.1 Sistem Pakar

Sistem pakar adalah kerangka kerja yang direncanakan dan dijalankan dengan bantuan bahasa program tertentu untuk memiliki opsi untuk menganani masalah seperti yang diselesaikan oleh para ahli. Sistem ini diharapkan orang awam dapat menyelesaikan masalah tertentu mulai berbagai masalah rumit sekalipun tanpa menggunakan pakar ahli dibidang tersebut. Sehubungan dengan specialist, kerangka kerja ini dapat digunakan sebagai asisten yang berpengalaman (Rohman & Fauzijah.2008). Sistem Pakar (Expert System) yaitu program berbasis informasi yang memberikan rkualitas untuk masalah pada bidang tertentu. Sistem pakar ialah program berbasis informasi yang memberikan jawaban berkualitas untuk dimasalah ruang tertentu. Sistem pakar adalah spesialis program PC yang

mencerminkan cara berpikir dan informasi tentang spesialis dalam menangani suatu masalah tertentu (Kusumadewi dalam Rohman & Fauziah, 2008:2). Jadi secara keseluruhan sangat mungkin beralasan bahwa sistem pakar adalah kerangka kerja yang mencoba untuk menerapkan informasi manusia ke PC sehingga PC dapat menangani masalah seperti yang biasanya dilakukan oleh spesialis untuk menangani masalah tertentu.

2.2.2 Diagnosis

Tindakan dalam hal menangani ketidakteraturan sesuatu menggunakan alat-alat bantu atau sistem untuk menghasilkan informasi dan membuat kesimpulan yang mungkin ketidakteraturan itu sendiri. Diagnosis sistem pakar biasanya digunakan untuk merekomendasikan suatu tindakan seperti diagnosis penyakit, kerusakan mesin, dan sebagainya. Diagnosa adalah istilah yang diadopsi dari bidang kedokteran dan kedokteran sebagai proses penentuan sifat suatu penyakit dari gejala yang ditimbulkannya. Dalam dunia pendidikan, istilah “diagnosis” merupakan istilah yang relatif baru dan sejalan (Poerwaarminto.2007). Diagnosis berarti mengidentifikasi suatu penyakit dengan cara mengamati atau meneliti gejala penyakit tersebut. Istilah ini biasa digunakan dalam kedokteran. Dalam dunia pendidikan, penyebab, jenis, dan jenis gangguan belajar pada siswa. Yang dimaksud dengan “diagnosis”, didefinisikan sebagai upaya untuk mengidentifikasi dan menyelidiki suatu fitur, tidak banyak berubah 19. Diagnosis adalah istilah yang kami gunakan. Bidang medis. diagnosis dapat didefinisikan sebagai Upaya atau proses untuk mengetahui kelemahan atau penyakit apa yang dimiliki seseorang melalui pengujian dan penelitian gejala yang cermat (Muhibbin, 2002).

2.2.3 Android

Android adalah kumpulan perangkat lunak untuk perangkat seluler ini termasuk sistem operasi, middleware, dan aplikasi seluler utama. Android memiliki empat karakteristik yaitu:

1. Terbuka

Android dibuat untuk sepenuhnya terbuka, sehingga aplikasi dapat menjalankan salah satu fungsi inti ponsel, seperti melakukan panggilan, mengirim pesan teks, menggunakan kamera, dan banyak lagi. Android menggunakan mesin virtual yang dirancang untuk mengoptimalkan memori dan sumber daya perangkat keras yang terdapat dalam perangkat.

Android adalah open source, dan dapat diperluas secara gratis untuk menyertakan teknologi baru yang lebih maju seiring munculnya teknologi ini. Platform akan terus berkembang untuk membangun aplikasi seluler yang inovatif.

2. Semua aplikasi dibuat sama

Android tidak memberikan perbedaan terhadap aplikasi pertama dari sebuah telepon selular dan aplikasi pihak ketiga (Third-party application). Semua aplikasi dapat dikembangkan untuk memiliki akses yang sama pada kemampuan sebuah telepon untuk menyediakan layanan dan aplikasi yang luas terhadap para pengguna.

3. Pengembangan aplikasi yang cepat

Android memberi pengguna berbagai akses yang sangat luas ke library dan alat yang mereka butuhkan untuk membangun aplikasi yang lebih baik. Android memiliki banyak alat yang dapat digunakan oleh developer untuk memaksimalkan daya produksi mereka saat merancang dan membangun aplikasi Google Inc. Android dirancang sepenuhnya open source sehingga developer bisa menggunakan android tanpa harus membayar lisensi dari Google untuk mengembangkan android tanpa adanya batasan.

2.2.4 Flutter

Flutter adalah SDK untuk kemajuan aplikasi serbaguna dengan eksekusi superior, aplikasi untuk iOS dan Android, dari basis kode (code base) yang dibuat oleh Google dengan izin sumber terbuka. Untuk membuat aplikasi Flutter, penting untuk menggunakan Dart. Dart adalah bahasa pemrograman dibuat oleh Google untuk menggantikan Javascript.

2.2.5 Vscode

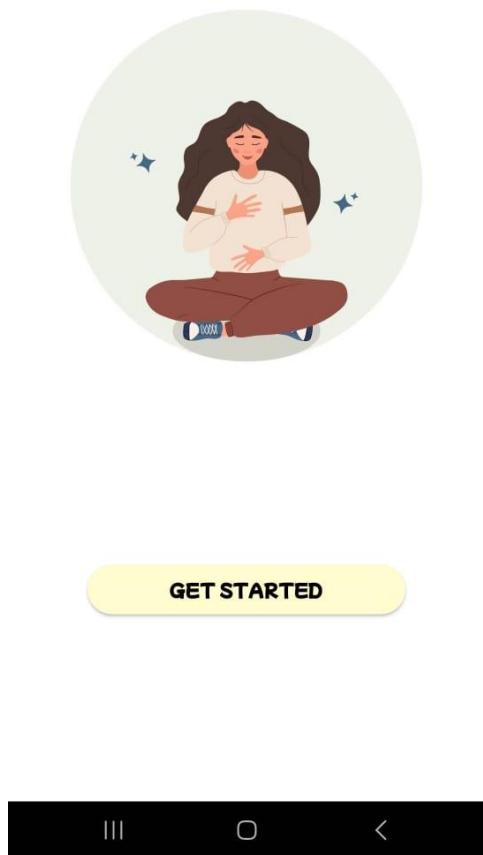
Visual Studio Code adalah aplikasi code editor buatan Microsoft yang dapat dijalankan di semua perangkat desktop secara gratis. Kelengkapan fitur dan ekstensi membuat code editor ini menjadi pilihan utama para pengembang. Visual Studio Code bahkan mendukung hampir semua sistem operasi seperti Windows, Mac OS, Linux, dan lain sebagainya.

BAB III

STRUKTUR MODUL DAN LOGIKA

3.1 STRUKTUR LOGIKA PROGRAM

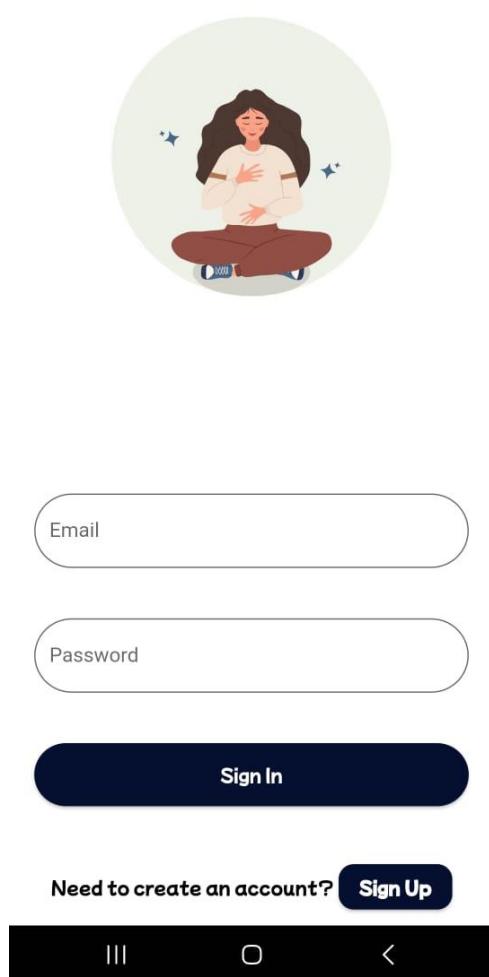
A. Output Modul 1 HomeScreen



Pada gambar HomeScreen menampilkan karakter wanita dengan rambut panjang, mata tertutup, tersenyum, dan tangannya berada di atas dada, menciptakan kesan bahwa dia sedang bersyukur atau merasakan sesuatu yang mendalam. Di sekitarnya ada bintang-bintang kecil yang melambangkan kebahagiaan atau energi positif.

Di bawah ilustrasi ada tombol dengan teks "GET STARTED" yang menunjukkan layar pembuka atau intro untuk aplikasi, dan mengklik tombol tersebut akan memulai proses penggunaan aplikasi atau layanan yang ditawarkan.

B. Output Modul 1 Login & Register



Merupakan salah satu modul yang banyak dibutuhkan dalam berbagai program, karena berguna untuk proses autentifikasi data user. Proses ini digunakan untuk mengakses aplikasi dengan memasukkan identitas dari akun pengguna dan password guna mengakses data login.

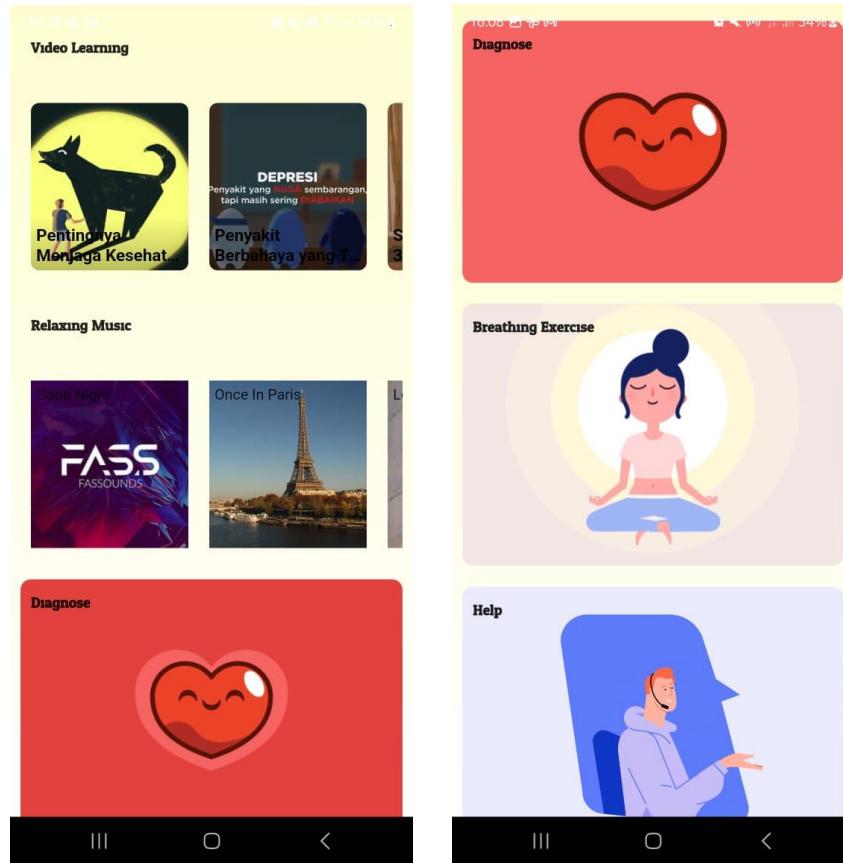
Register merupakan tahapan yang digunakan untuk mendaftarkan informasi user. Di bagian paling bawah, ada teks "Need to create an account?" dengan opsi "Sign Up" yang diberi tautan, menunjukkan bahwa jika pengguna belum memiliki akun, mereka dapat membuatnya dengan mengklik "Sign Up". Ini adalah praktik desain standar untuk halaman login, dimana menyediakan kedua opsi untuk masuk dan mendaftar dalam satu tampilan untuk memudahkan navigasi pengguna.

C. Output Modul Welcome Page



Modul ini merupakan modul halaman utama di mana dalam modul ini menunjukkan bahwa pengguna telah masuk ke dalam sistem dan telah dikenali oleh aplikasi dengan nama atau username "Lyswty". Pesan "Let's manage your mental with us!" mengindikasikan bahwa aplikasi ini menawarkan layanan atau alat untuk membantu pengguna mengelola kesehatan mental mereka.

D. Output Modul Main Menu

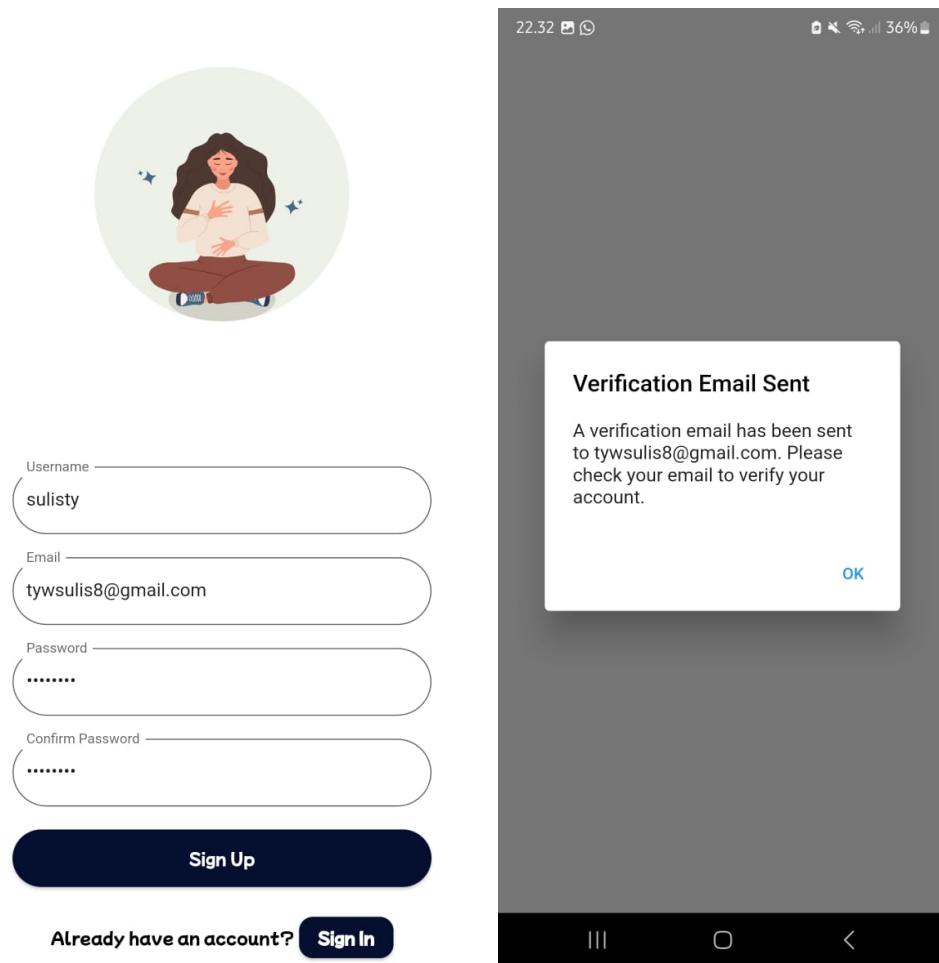


Modul Main Menu merupakan modul utama yang berisi daftar perintah-perintah pada program, yang apabila dieksekusi akan menjalankan suatu perintah tertentu dari program tersebut. Modul tampilan utama kami buat untuk memenuhi spesifikasi program terkait. Pada modul ini tampilan utama diatas yang terdiri dari :

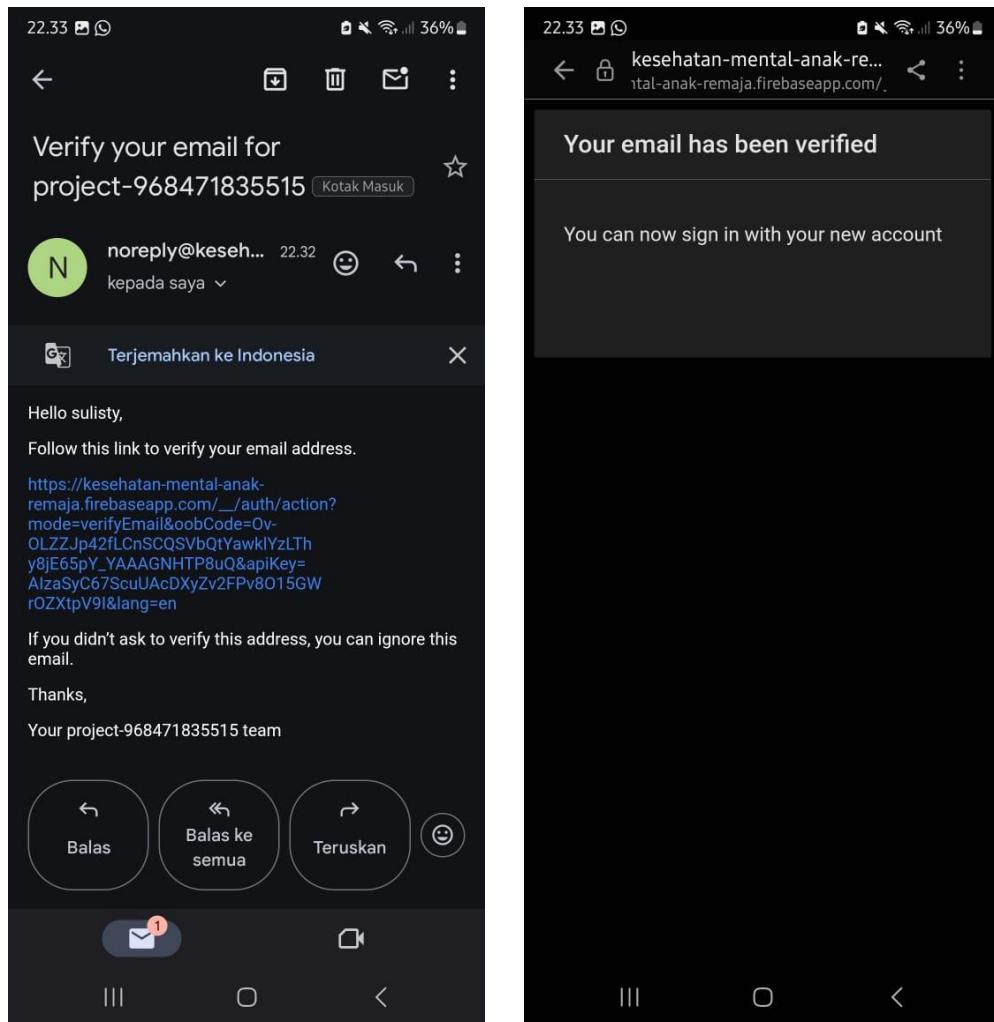
- a) Vidio Learning : merupakan video edukasi tentang kesehatan mental
- b) Relaxing Music : menawarkan musik relaksasi untuk membantu pengguna mengurangi stres dan meningkatkan kesejahteraan mental
- c) Diagnose : adalah bagian interaktif untuk melakukan "self-check" atau penilaian mandiri kesehatan mental pengguna
- d) Breathing Exercise : menyediakan panduan atau latihan pernapasan yang bertujuan untuk membantu pengguna relaksasi dan mengelola stress
- e) Help : untuk mendapatkan bantuan lebih lanjut

E. Test Modul User

1) Test Modul 1 Register & Login (User)

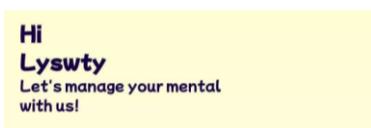


- ❖ Modul Login dan register ini dibuat untuk mengakses aplikasi dengan memasukkan identitas dari akun user. Untuk user yang belum melakukan register diharuskan melakukan register terlebih dahulu. Untuk daftar akun atau register user harus mengisi username, email, dan password pada form pendaftaran aplikasi dan menekan tombol "Sign Up" untuk membuat akun baru.
- ❖ Setelah menekan "Sign Up", sistem aplikasi mengirimkan email verifikasi ke alamat email yang dimasukkan pengguna (dalam kasus ini tywsulis8@gmail.com) untuk memastikan bahwa email tersebut valid dan dimiliki oleh pengguna.
- ❖ Sebuah pop-up muncul di aplikasi yang memberitahu pengguna bahwa email verifikasi telah dikirim ke alamat email mereka, dengan instruksi untuk memeriksa email mereka dan mengklik link verifikasi.



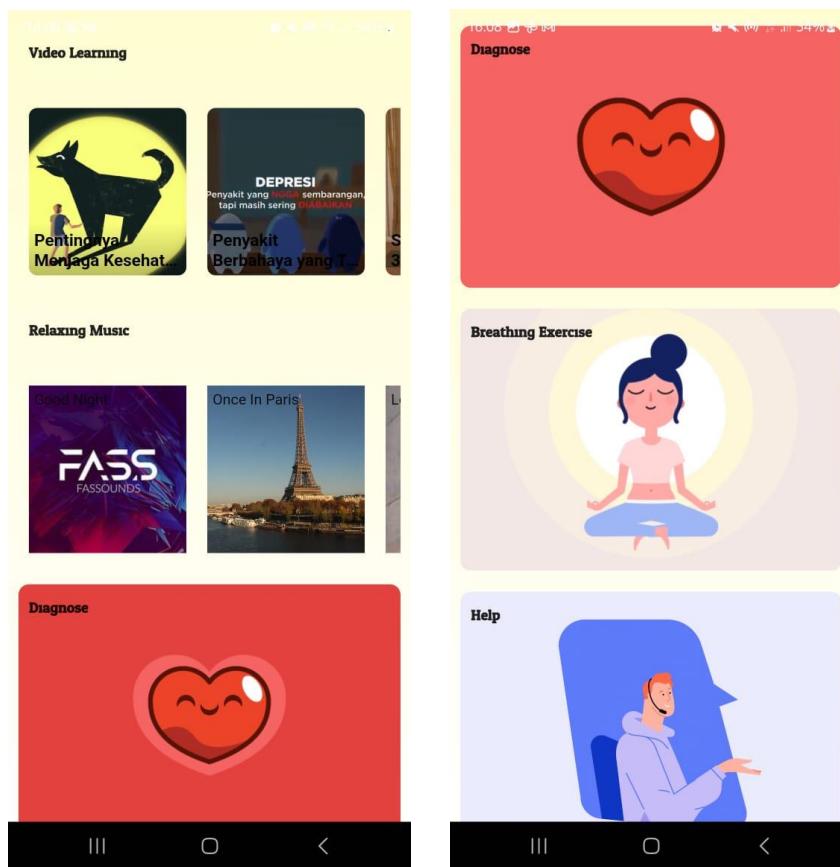
- ❖ Pengguna menerima email dari sistem dengan subjek "Verify your email for project-968471835515" yang berisi link untuk verifikasi. Pengguna diminta untuk mengklik link tersebut untuk menyelesaikan proses verifikasi akun mereka.
- ❖ Setelah pengguna mengklik link verifikasi, mereka diarahkan ke halaman yang memberitahu mereka bahwa email mereka telah diverifikasi dengan sukses.
- ❖ Halaman verifikasi juga memberikan informasi bahwa pengguna sekarang dapat masuk (sign in) ke dalam aplikasi dengan akun baru yang telah diverifikasi tersebut.

2) Test Modul Welcome Page

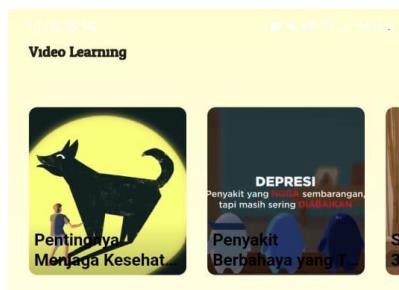


Modul Welcome Page akan muncul jika user pertama kali login. Modul ini dibuat untuk tampilan selamat datang sebelum Aplikasi berjalan. Tujuannya agar user mengetahui program yang akan segera dibuka adalah Aplikasi dari program Kesehatan Mental. Pada tampilan ini terdapat beberapa dialog seperti “Hi, User”

3) Test Modul Main Menu

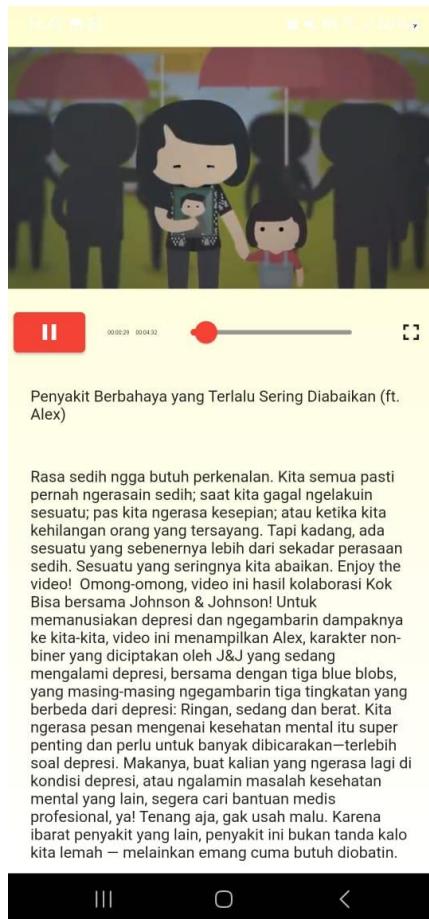


a) Video Learning



Bagian ini merupakan tampilan video edukasi tentang kesehatan mental.

Apabila user mengklik salah satu dari vidio Learning yang tersedia , maka program akan menampilkan seperti gambar di bawah ini :



Deskripsi Video: Deskripsi tersebut menyentuh tentang perasaan sedih yang umum dialami oleh banyak orang dan menekankan bahwa sedih adalah sesuatu yang lebih dari sekadar perasaan sementara. Video ini bertujuan untuk menyadarkan penonton mengenai pentingnya mengenali dan mengatasi depresi, yang digambarkan sebagai "penyakit berbahaya yang seringkali diabaikan."

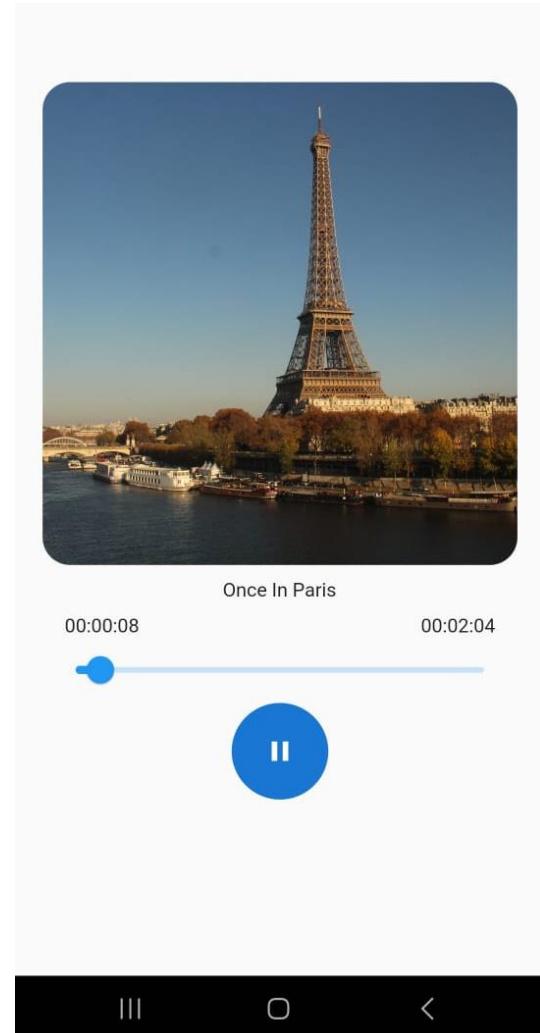
User dapat mengklik tombol play/pause, timeline video yang menunjukkan durasi video dan seberapa jauh video tersebut telah diputar.

b) Relaxing Music

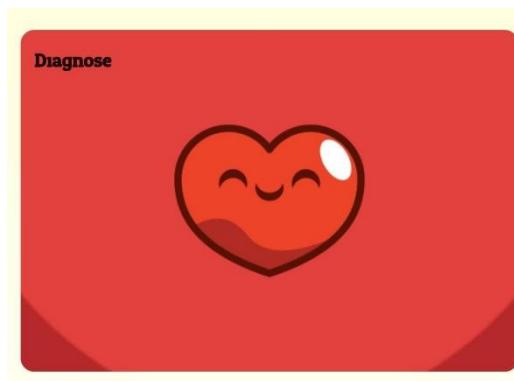


Bagian ini merupakan tampilan Relaxing Music yang menawarkan musik relaksasi untuk membantu pengguna dalam mengurangi stress

Apabila user mengklik salah satu dari Musik Relaksasi yang tersedia , maka program akan menampilkan seperti gambar di bawah ini :



c) Diagnose



Merupakan bagian interaktif untuk melakukan "self-check" atau penilaian mandiri kesehatan mental pengguna dengan mengisi quisioner. Apabila user mengklik maka program akan menampilkan kumpulan quisioner yang harus di isi oleh user seperti gambar di bawah ini :

16.52

Pikiran untuk menyakiti diri sendiri, perilaku merusak diri

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Next

16.53

kelelahan tidur atau tidur berlebihan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.53

penurunan atau kenaikan berat badan secara signifikan tanpa alasan jelas.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.53

Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.54

kesulitan untuk fokus, mudah terdistraksi

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.54

menghindari interaksi sosial, menyendiri

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.53 ☺ 51%  51% 

merasa putus asa dan hilang harapan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.54 ☺ 50%  50% 

pola makan tidak sehat, obsesi dengan berat badan

sering sekali
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.54 ☺  50%  50% 

merasa sulit meningkatkan inisiatif dalam melakukan sesuatu

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.55 ☺  50%  50% 

Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.53 ☺ 51%  51% 

Tidak dapat melihat hal positif dari suatu kegiatan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.54 ☺  50%  50% 

Perasaan tidak berharga atau bersalah tanpa alasan yang jelas.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

[Prev](#) [Next](#)

16.55 ☰ 🔍 50%

Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.54 ☰ 🔍 50%

merasa tidak ada hal yang diharapkan di masa depan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.55 ☰ 🔍 50%

kelelahan berkelanjutan, kurang motivasi

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.55 ☰ 🔍 50%

reaksi emosional yang berlebihan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Submit**

Setelah semua quisioner sudah terisi , user dapat mengklik tombol submit dan selanjutnya program akan menampilkan hasil diagnose dari quisioner yang sebelumnya di isi oleh user , untuk output nya seperti gambar di bawah ini

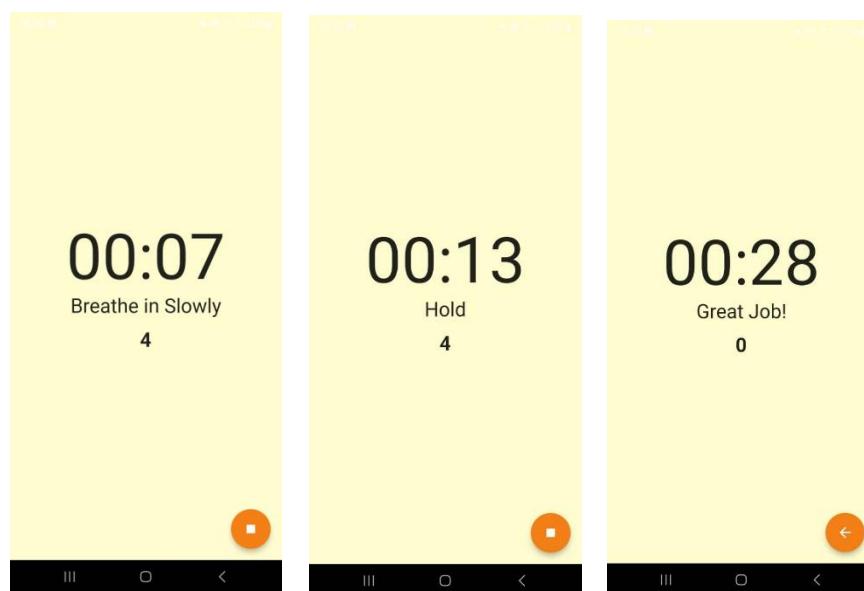
Output :

Result Mental Health Report: Your Level Of Depression: 65%	
<p>Question: Pikiran untuk menyakiti diri sendiri, perilaku merusak diri Answer: lumayan sering</p> <p>Question: kelelahan tidur atau tidur berlebihan Answer: kadang-kadang</p> <p>Question: Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah. Answer: kadang-kadang</p> <p>Question: penurunan atau kenaikan berat badan secara signifikan tanpa alasan jelas. Answer: tidak pernah</p> <p>Question: Tidak dapat melihat hal positif dari suatu kegiatan Answer: kadang-kadang</p> <p>Question: merasa putus asa dan hilang harapan Answer: kadang-kadang</p> <p>Question: kesulitan untuk fokus, mudah terdistraksi Answer: lumayan sering</p> <p>Question: Perasaan tidak berharga atau bersalah tanpa alasan yang jelas. Answer: kadang-kadang</p> <p>Question: pola makan tidak sehat, obsesi dengan berat badan Answer: kadang-kadang</p> <p>Question: merasa tidak ada hal yang diharapkan di masa depan</p>	<p>Answer: kadang-kadang</p> <p>Question: merasa tidak ada hal yang diharapkan di masa depan Answer: kadang-kadang</p> <p>Question: menghindari interaksi sosial, menyendiri Answer: kadang-kadang</p> <p>Question: merasa sulit meningkatkan inisiatif dalam melakukan sesuatu Answer: kadang-kadang</p> <p>Question: kelelahan berkelanjutan, kurang motivasi Answer: kadang-kadang</p> <p>Question: Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas. Answer: kadang-kadang</p> <p>Question: Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih. Answer: lumayan sering</p> <p>Question: reaksi emosional yang berlebihan Answer: lumayan sering</p>
<p>Types of Symptoms: Depresi Berat Tanpa Gangguan Fungsional</p> <p>Detail Symptoms and recommendations Kekhawatiran dan perubahan mood: Rekomendasikan konsultasi dengan psikolog atau terapi perilaku kognitif dan Pertimbangkan evaluasi untuk gangguan mood dan terapi stabilisasi mood. Pola Makan Tidak Sehat: disarankan konsultasi dengan ahli gizi dan terapi perilaku. Pikiran untuk Menyakiti Diri Sendiri: Segera rekomendasikan bantuan profesional dan pertimbangkan manajemen krisis.</p>	
<input type="button" value="Save Diagnose"/>	
III □ <	III □ <

- Level of Depression: 65%: Ini menunjukkan tingkat depresi yang dihitung oleh aplikasi berdasarkan jawaban yang diberikan. Angka 65% mungkin menunjukkan risiko atau tingkat depresi yang moderat.
- Pertanyaan dan Jawaban: Ada serangkaian pertanyaan yang terkait dengan gejala depresi. Setiap pertanyaan memiliki jawaban yang beragam, seperti "kadang-kadang" atau "lumayan sering", yang mungkin digunakan untuk menilai frekuensi atau intensitas gejala yang dialami oleh pengguna.
- Types of Symptoms: Depresi Berat Tanpa Gangguan Fungsional: Ini mengindikasikan bahwa berdasarkan jawaban yang diberikan, gejala yang dilaporkan cocok dengan apa yang disebut sebagai "Depresi Berat Tanpa Gangguan Fungsional". Ini mungkin berarti bahwa walaupun gejala depresi yang serius hadir, pengguna masih dapat menjalankan fungsi sehari-hari mereka.

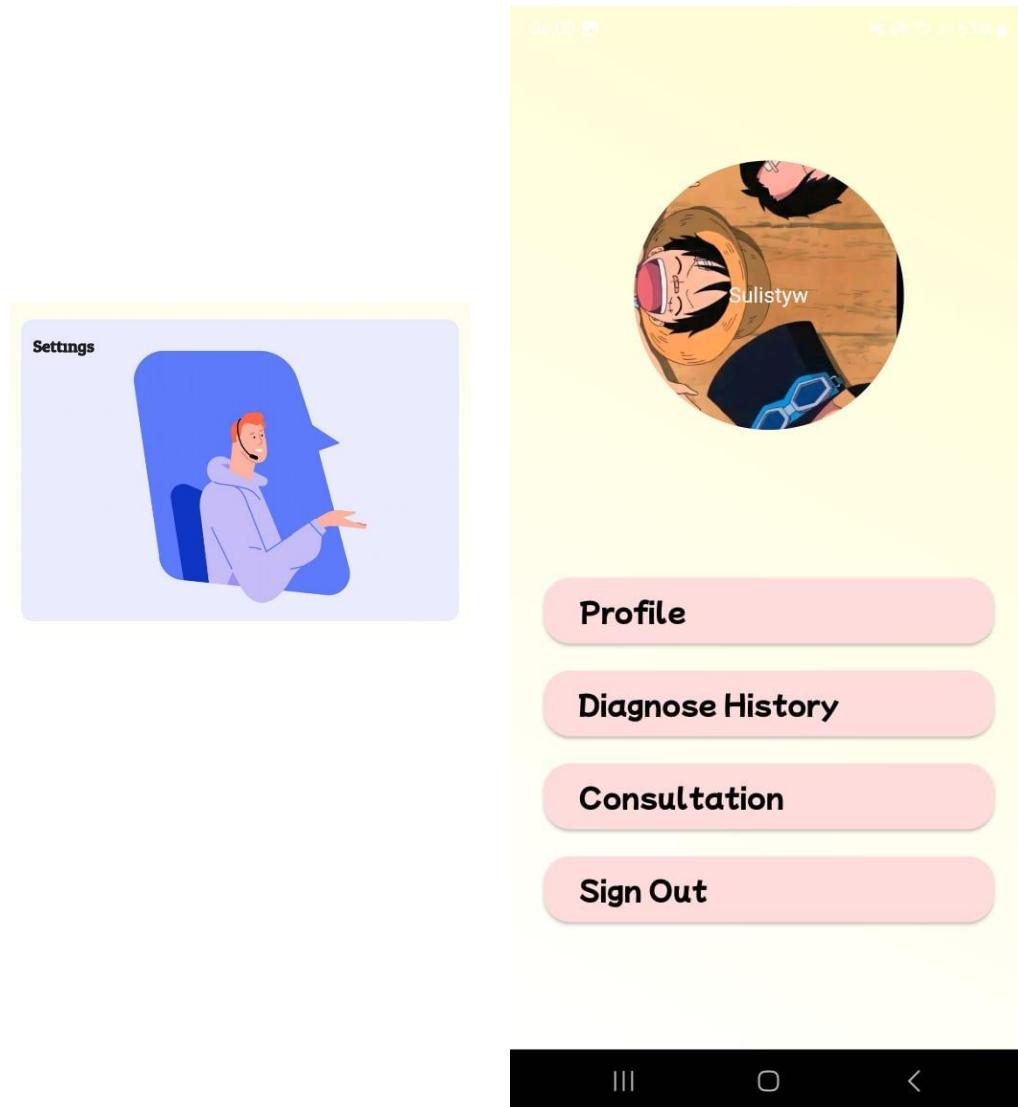
- Detail Symptoms and Recommendations: Bagian ini memberikan ringkasan tentang gejala yang dialami dan memberikan beberapa rekomendasi umum. Rekomendasi termasuk konsultasi dengan psikolog atau psikiater untuk terapi perilaku kognitif dan terapi stabilisasi mood. Juga disarankan untuk mengevaluasi penggunaan obat-obatan dan mempertimbangkan manajemen krisis jika perlu.
- Save Diagnose: Tombol ini tampaknya memungkinkan pengguna untuk menyimpan diagnosis yang diberikan oleh aplikasi.

d) Breathing Exercise



- Tampilan di atas menunjukkan bahwa aplikasi ini memberikan panduan untuk latihan pernapasan, yang bisa membantu pengguna untuk rileks atau mengelola stres.
- Menampilkan timer yang sedang menghitung mundur dari 7 detik dengan instruksi "Breathe in Slowly 4". Ini menandakan bahwa pengguna harus menghirup napas secara perlahan selama 4 hitungan.
- Timer menunjukkan hitungan mundur dari 13 detik dengan instruksi "Hold 4", yang berarti pengguna diharapkan menahan napas selama 4 hitungan.
- Gambar Keempat (kanan bawah): Timer ini menunjukkan hitungan mundur dari 28 detik dengan pesan "Great Job! 0". Ini menandakan bahwa siklus latihan pernapasan telah selesai, dan pengguna telah menyelesaikan satu ronde latihan pernapasan.

e) Settings



Bagian ini menampilkan sebuah ilustrasi yang menggambarkan 'Settings' atau pengaturan aplikasi. Ikon pengaturan menunjukkan bahwa pengguna dapat mengakses berbagai opsi untuk mengkonfigurasi aplikasi sesuai dengan preferensi mereka.

- **Profile:** Tempat pengguna dapat melihat dan mengedit informasi profil mereka.
- **Diagnose History:** Sebuah fitur yang tampaknya menyimpan riwayat diagnosa atau hasil kesehatan yang telah dilakukan sebelumnya.
- **Consultation:** Bagian di mana pengguna dapat menjadwalkan atau melihat riwayat konsultasi medis.
- **Sign Out:** Opsi untuk keluar dari akun pengguna di aplikasi.

F. Test Modul Admin

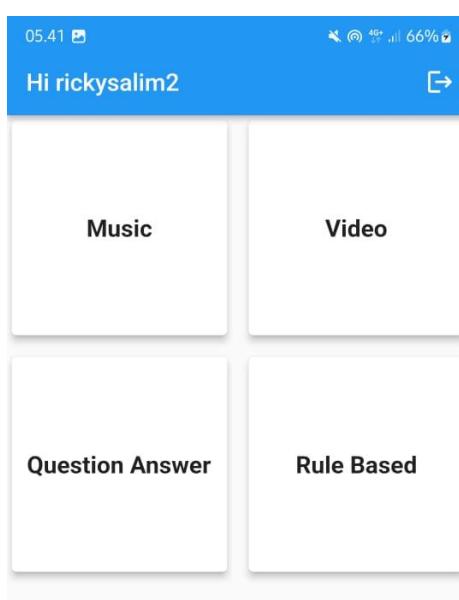
1. Test Modul 1 Login



untuk halaman login admin dari sebuah aplikasi. Pada halaman ini, terdapat dua bidang yang harus diisi:

- ❖ Email: Pengguna harus memasukkan alamat email mereka yang terkait dengan akun admin. Dalam gambar ini, alamat email 'rickysalim71@gmail.com' sudah terisi, yang mungkin adalah contoh atau input sebelumnya.
- ❖ Password: Pengguna harus memasukkan kata sandi yang sesuai dengan alamat email mereka.

2. Test Modul Main Menu



Pada Tampilan utama (main menu) dari sebuah antarmuka admin pada aplikasi. Tampilan ini menampilkan berbagai opsi yang dapat diakses oleh admin:

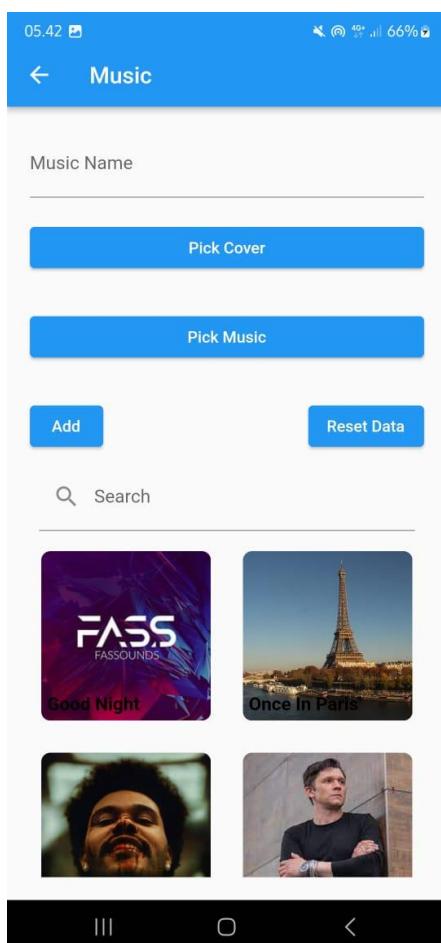
- Music: Kemungkinan ini adalah bagian dimana admin dapat mengelola konten musik, seperti menambahkan lagu baru, mengedit informasi lagu, atau menghapus entri.
- musik, tetapi untuk pengelolaan konten video.

- Question Answer: Merujuk pada modul untuk mengelola sesi tanya jawab, FAQ, atau bahkan bagian untuk asisten virtual yang berbasis pada pertanyaan dan jawaban.
- Rule Based: Opsi ini terkait dengan pengaturan aturan-aturan atau algoritma yang digunakan dalam aplikasi atau situs web, seperti aturan untuk moderasi konten atau aturan bisnis lainnya.

Di bagian atas layar, terdapat sapaan "Hi rickysalim2", yang menunjukkan bahwa sistem telah mengenali pengguna yang masuk dan menampilkan nama pengguna atau ID mereka.

Tampilan menu ini dirancang untuk memberikan akses cepat ke fitur-fitur penting yang perlu dikelola oleh admin, dengan tampilan yang bersih dan terorganisir untuk memudahkan navigasi dan penggunaan.

3. Test Modul Main Menu (Music)



Tampilan dari bagian "Music" dalam antarmuka admin aplikasi. Fitur-fitur yang ditampilkan adalah:

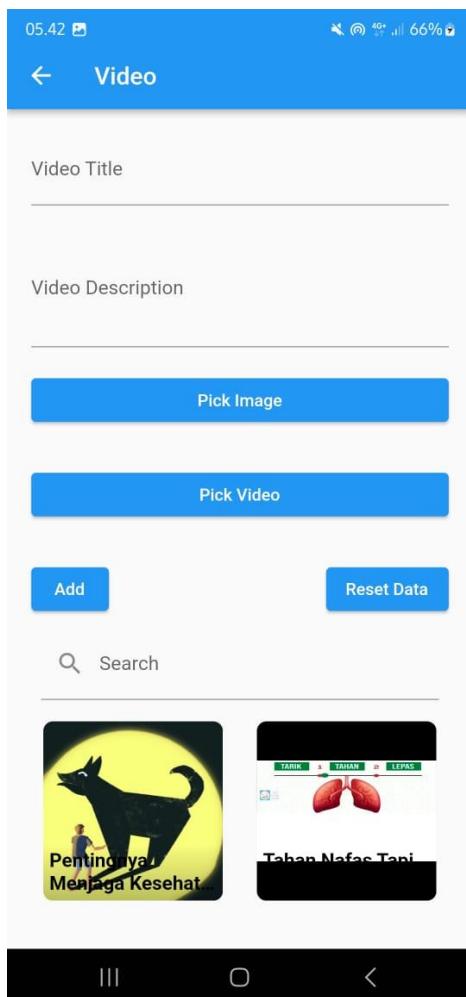
- Music Name: Sebuah bidang teks tempat admin dapat memasukkan nama lagu yang akan ditambahkan atau dikelola.
- Pick Cover: Tombol ini digunakan untuk memilih atau mengunggah gambar sampul untuk lagu tersebut.
- Pick Music: Tombol ini digunakan untuk memilih atau mengunggah file musik yang sesuai.
- Add: Setelah memilih sampul dan musik serta memasukkan nama, admin dapat menambahkan lagu ke dalam sistem dengan tombol "Add".

- Reset Data: Tombol ini kemungkinan digunakan untuk menghapus input yang telah dimasukkan tanpa menyimpannya.
- Search: Fitur pencarian memungkinkan admin untuk mencari lagu yang sudah ada dalam database.

Di bawah fitur pencarian, terdapat thumbnail dari lagu yang mungkin sudah ditambahkan ke dalam sistem, dengan contoh lagu "Good Night" dan "Once In Paris", yang masing-masing memiliki gambar sampulnya sendiri.

Antarmuka ini dirancang untuk memudahkan admin dalam mengelola konten musik, dari menambahkan lagu baru, mengedit informasi yang ada, hingga mengatur sampul dan file musik yang terkait dengan setiap entri.

4. Test Modul Main Menu (Vidio)



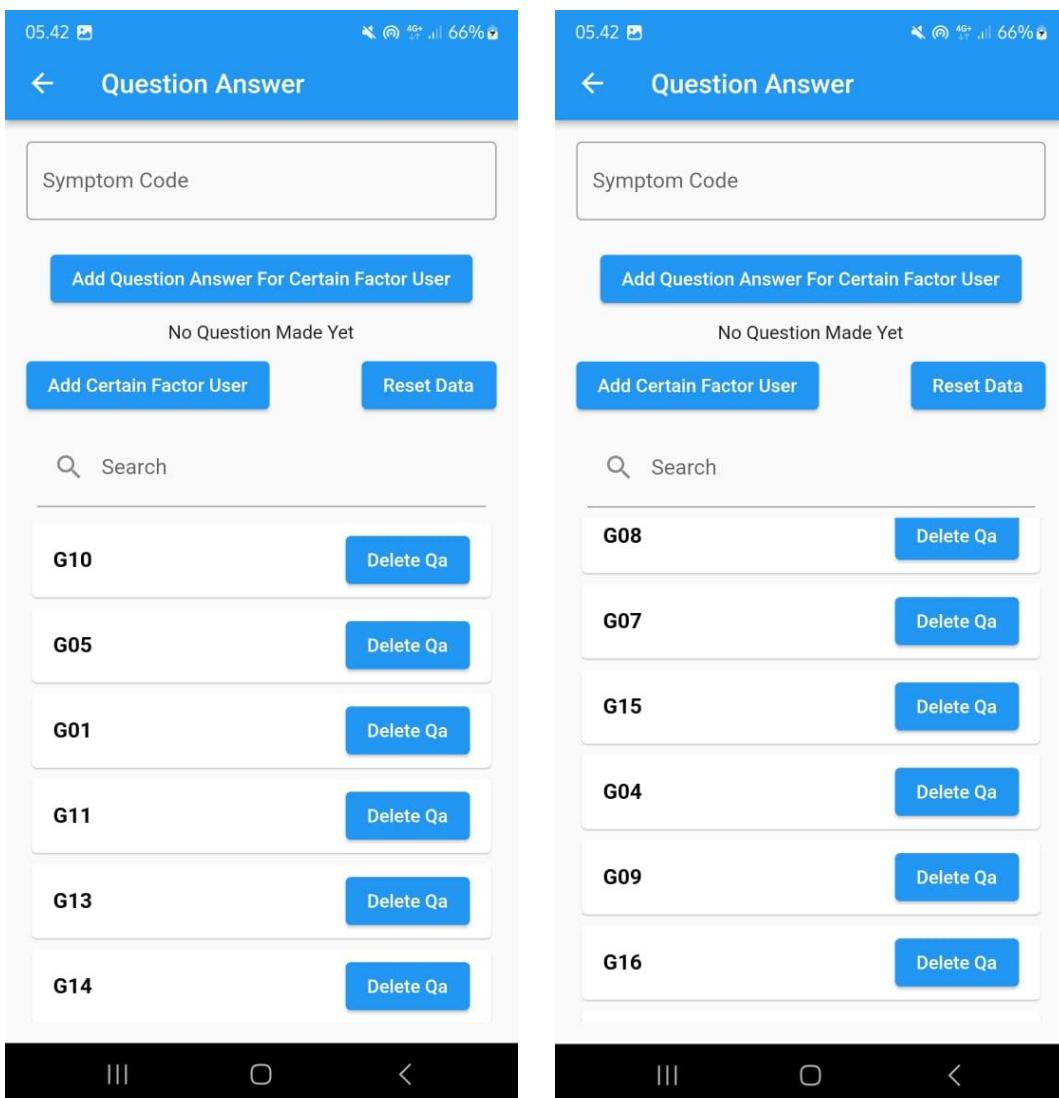
Tampilan menu utama untuk bagian "Video" dari sebuah antarmuka admin aplikasi atau situs web.

Berikut ini adalah fitur-fitur yang ditampilkan:

- ❖ Video Title: Sebuah bidang input di mana admin dapat memasukkan judul video yang akan ditambahkan atau dikelola.
- ❖ Video Description: Sebuah bidang input untuk menambahkan deskripsi video, yang mungkin digunakan untuk memberikan konteks atau informasi tambahan tentang video tersebut.
- ❖ Pick Image: Tombol ini digunakan untuk memilih atau mengunggah gambar thumbnail yang akan ditampilkan bersama video di aplikasi atau situs web.
- ❖ Pick Video: Tombol ini digunakan untuk memilih atau mengunggah file video yang ingin ditambahkan ke dalam sistem.

- ❖ Add: Setelah memilih thumbnail dan video serta memasukkan judul dan deskripsi, admin dapat menggunakan tombol "Add" untuk menambahkan video ke dalam sistem.
- ❖ Reset Data: Tombol ini mungkin digunakan untuk menghapus semua input yang telah dimasukkan tanpa menyimpannya.
- ❖ Search: Sebuah fitur pencarian yang memungkinkan admin untuk mencari video yang sudah ada dalam database.

5. Test Modul Main Menu (Question Answer)



Fitur-fitur yang terlihat adalah:

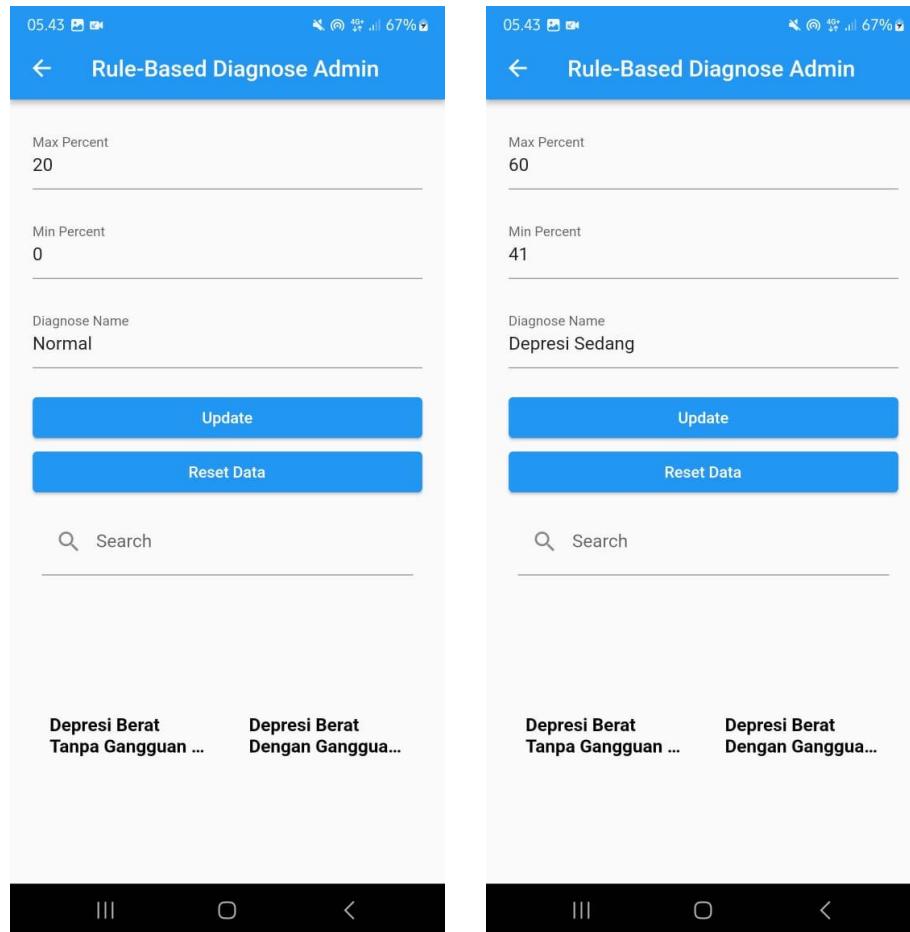
- Symptom Code: Sebuah bidang teks tempat admin dapat memasukkan kode untuk gejala tertentu.

- Add Question Answer For Certain Factor User: Tombol ini digunakan untuk menambahkan pasangan pertanyaan dan jawaban baru yang terkait dengan faktor pengguna tertentu yang telah didefinisikan oleh sistem atau aplikasi.
- No Question Made Yet: Ini mungkin merupakan pesan status yang menunjukkan belum adanya pertanyaan yang dimasukkan untuk kode gejala tertentu.
- Add Certain Factor User: Tombol ini mungkin untuk menambahkan faktor pengguna baru ke dalam sistem, yang bisa berhubungan dengan variabel yang mempengaruhi gejala atau kondisi kesehatan.
- Reset Data: Tombol ini kemungkinan untuk menghapus semua input di bidang teks tanpa menyimpannya.
- Search: Sebuah fitur pencarian yang memungkinkan admin untuk mencari kode gejala atau pertanyaan yang sudah ada dalam database.

Di bawah fitur pencarian, terdapat daftar kode gejala (misalnya, G10, G05, G01, dll.) dengan opsi untuk menghapus pertanyaan yang terkait dengan kode tersebut menggunakan tombol "Delete Qa".

6. Test Modul Main Menu (Rule Based Diagnose)

<div style="background-color: #0072BD; color: white; padding: 5px; text-align: center;"> 05.43 67% </div> <div style="background-color: #0072BD; color: white; padding: 5px; margin-bottom: 5px;"> ← Rule-Based Diagnose Admin </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Max Percent 100 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Min Percent 81 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Diagnose Name Depresi Berat Dengan Gangguan Fungsional </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Update </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Reset Data </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Search </div> <div style="display: flex; justify-content: space-around; font-size: 0.8em; margin-top: 20px;"> Depresi Berat Tanpa Gangguan ... Depresi Berat Dengan Ganggu... </div>	<div style="background-color: #0072BD; color: white; padding: 5px; text-align: center;"> 05.43 67% </div> <div style="background-color: #0072BD; color: white; padding: 5px; margin-bottom: 5px;"> ← Rule-Based Diagnose Admin </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Max Percent 80 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Min Percent 61 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Diagnose Name Depresi Berat Tanpa Gangguan Fungsional </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Update </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Reset Data </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Search </div> <div style="display: flex; justify-content: space-around; font-size: 0.8em; margin-top: 20px;"> Depresi Berat Tanpa Gangguan ... Depresi Berat Dengan Ganggu... </div>	<div style="background-color: #0072BD; color: white; padding: 5px; text-align: center;"> 05.43 67% </div> <div style="background-color: #0072BD; color: white; padding: 5px; margin-bottom: 5px;"> ← Rule-Based Diagnose Admin </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Max Percent 40 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Min Percent 21 </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Diagnose Name Depresi Ringan </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Update </div> <div style="background-color: #0072BD; color: white; padding: 5px; text-align: center; margin-bottom: 5px;"> Reset Data </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Search </div> <div style="display: flex; justify-content: space-around; font-size: 0.8em; margin-top: 20px;"> Depresi Berat Tanpa Gangguan ... Depresi Berat Dengan Ganggu... </div>
---	---	--



"Rule-Based Diagnose Admin", yang nampaknya digunakan untuk mengatur aturan-aturan dalam diagnosa berbasis kriteria tertentu. Berikut adalah penjelasan dari setiap gambar:

- Gambar Pertama: Menunjukkan sebuah pengaturan untuk diagnosis dengan nama "Normal", dengan persentase maksimal 20% dan minimal 0%. Ini bisa berarti bahwa rentang persentase ini digunakan untuk mengklasifikasikan hasil diagnosa sebagai "Normal".
- Gambar Kedua: Menampilkan pengaturan untuk "Depresi Ringan", dengan rentang persentase dari 21% hingga 40%. Rentang ini kemungkinan digunakan untuk menentukan apakah hasil dari diagnosa sesuai dengan kriteria "Depresi Ringan".
- Gambar Ketiga: Memperlihatkan pengaturan untuk "Depresi Berat Dengan Gangguan Fungsional", dengan rentang persentase dari 81% hingga 100%. Ini menandakan bahwa jika hasil diagnosa jatuh dalam rentang ini, maka akan dikategorikan sebagai "Depresi Berat Dengan Gangguan Fungsional".
- Gambar Keempat: Menunjukkan pengaturan untuk "Depresi Berat Tanpa Gangguan Fungsional", dengan rentang persentase dari 61% hingga 80%

G. Pengujian Skenario

1) Pengujian Beta dengan User Acceptance Testing (UAT)

Tabel 3.1 Kategori Skor

No	Singkatan	Keterangan	Skor
1	SS	Sangat Setuju	4
2	S	Setuju	3
3	TS	Tidak Setuju	2
4	STS	Sangat Tidak Setuju	1

Tabel 3.2 Pernyataan Penilaian UAT

No	Keterangan	Skors			
		SS	S	TS	STS
<i>Usability</i>					
1	Aplikasi sistem pakar mudah dipelajari untuk dioperasikan				
2	Interaksi dengan Aplikasi Sistem Pakar jelas dan mudah dimengerti				
3	Saya merasa Aplikasi Sistem Pakar mudah untuk digunakan				
4	Website Sistem Pakar memiliki tampilan yang menarik				
5	aplikasi merespons ketika Anda melakukan tindakan tertentu				

No	Keterangan	Skors			
		SS	S	TS	STS
<i>Information</i>					
1	Aplikasi Sistem Pakar menyediakan informasi yang akurat				
2	Aplikasi Sistem pakar menyediakan informasi yang mudah dimengerti				
3	Aplikasi Sistem Pakar menyajikan informasi dalam yang sesuai				
4	Aplikasi ini membantu meningkatkan pemahaman Anda tentang kondisi kesehatan mental				

5	aplikasi memberikan informasi yang cukup untuk memahami kondisi kesehatan mental dengan baik				
---	--	--	--	--	--

No	Keterangan	Skors			
		SS	S	TS	STS
<i>Service Interaction</i>					
1	Aplikasi Sistem Pakar memberikan ruang untuk personalisasi				
2	Aplikasi Sistem Pakar memberikan layanan sesuai dengan apa yang disajikan				
3	Aplikasi Sistem Pakar memberikan kemudahan untuk menarik minat dan perhatian				
4	Fasilitas kontak darurat di aplikasi mudah diakses dan memberikan rasa keamanan.				
5	Layanan dukungan pengguna (helpdesk, pusat bantuan, dll.) mudah diakses.				

No	Keterangan	Skors			
		SS	S	TS	STS
<i>User Satisfaction</i>					
1	Apakah anda merasa puas terhadap kualitas penggunaan pada aplikasi ini				
2	Apakah anda merasa puas terhadap kualitas informasi pada aplikasi system pakar				
3	Apakah anda merasa puas terhadap kualitas interaksi/layanan pada aplikasi				

Tabel 3.3 Acuan Penilaian

Kriteria (%)	Keterangan
81-100%	Sangat Setuju
61-80%	Setuju
41-60%	Tidak Setuju
<=40%	Sangat Tidak Setuju

Perhitungan persentase pada kuesioner yang diperoleh untuk menghasilkan yang sesuai dilakukan dengan memakai rumus yaitu sebagai berikut :

$$P = \frac{\Sigma R}{N} \times 100\%$$

Keterangan:

P = Persentase skor yang dicari

ΣR = Jumlah jawaban

N = Jumlah skor maksimal

Perhitungan Persentase:

$$\text{Persen \%} = \frac{\text{Persentase skor yang dicari}}{\text{Jumlah skor maksimal}} \times 100\%$$

$$\text{Contoh Persen \%} = \frac{132}{148} \times 100 = 89,18\%$$

Tabel 3.4 Hasil Semua Quisoner UAT

No	Pertanyaan	Total Skor	Rata-rata	%	Ket
<i>Usability</i>					
1	Aplikasi sistem pakar mudah dipelajari untuk dioperasikan	132	3.56	89.18%	Sangat Setuju
2	Interaksi dengan Aplikasi Sistem Pakar jelas dan mudah dimengerti	118	3.18	79.72%	Setuju
3	Saya merasa Aplikasi Sistem Pakar mudah untuk digunakan	130	3.51	87.83%	Sangat Setuju
4	Website Sistem Pakar memiliki tampilan yang menarik	123	3.32	83.10%	Sangat Setuju
5	aplikasi merespons ketika Anda melakukan tindakan tertentu	117	3.16	79.05%	Setuju
Jumlah Total		620	16.73	418.88%	Sangat Setuju
Rata-Rata		124	3.346	83.78%	Sangat Setuju

<i>Information</i>					
1	Aplikasi Sistem Pakar menyediakan informasi yang akurat	111	3	75%	Setuju
2	Aplikasi Sistem pakar menyediakan informasi yang mudah dimengerti	114	3.08	77.02%	Setuju
3	Aplikasi Sistem Pakar menyajikan informasi dalam yang sesuai	124	3.35	83.78%	Sangat Setuju
4	Aplikasi ini membantu meningkatkan pemahaman Anda tentang kondisi kesehatan mental	125	3.37	84.45%	Sangat Setuju
5	aplikasi memberikan informasi yang cukup untuk memahami kondisi kesehatan mental dengan baik	121	3.27	81.75%	Sangat Setuju
Jumlah Total		595	16.07	402%	Setuju
Rata-Rata		119	3.214	80%	Setuju

<i>Service Interaction</i>					
1	Aplikasi Sistem Pakar memberikan ruang untuk personalisasi	121	3.27	81.75%	Sangat Setuju
2	Aplikasi Sistem Pakar memberikan layanan sesuai dengan apa yang disajikan	129	3.48	87.16%	Sangat Setuju
3	Aplikasi Sistem Pakar memberikan kemudahan untuk menarik minat dan perhatian	114	3.08	77.02%	Setuju
4	Fasilitas kontak darurat di aplikasi mudah diakses dan memberikan rasa keamanan.	135	3.64	91.21%	Sangat Setuju
5	Layanan dukungan pengguna (helpdesk, pusat bantuan, dll.) mudah diakses.	115	3.10	77.70%	Setuju
Jumlah Total		614	16.57	414.84%	Sangat Setuju
Rata-Rata		122.8	3.314	82.97%	Sangat Setuju

<i>User Satisfaction</i>					
1	Apakah anda merasa puas terhadap kualitas penggunaan pada aplikasi ini	128	3.45	86.48%	Sangat Setuju
2	Apakah anda merasa puas terhadap kualitas informasi pada aplikasi system pakar	120	3.24	81.08%	Sangat Setuju
3	Apakah anda merasa puas terhadap kualitas interaksi/layanan pada aplikasi	125	3.37	84.45%	Sangat Setuju
Jumlah Total		373	10.06	252.01%	Sangat Setuju
Rata-Rata		124.3	3.35	84.01%	Sangat Setuju

Berdasarkan pada tabel 4.9, dari 37 responden yang memberikan jawaban dengan menggunakan rumus yang tertera/digunakan oleh peneliti, didapatkan hasil sebanyak 82,65% untuk semua aspek pada metode Webqual dan variabel kapuasan pengguna. Peneliti mengambil kesimpulan bahwa Sistem Pakar Diagnosa Mental Illness memakai metode forward chaining sudah tergolong sangat baik/sangat layak.

BAB IV

HASIL DAN PEMBAHASAN

1. PEMBAGIAN KERJA DALAM KELOMPOK

Hari/Tanggal	Kegiatan
02/12/2023	Membahas Metode , UML pembuatan class diagram yang akan digunakan dalam sistem pakar diagnosis kesehatan mental serta membahasa konsep bentuk design aplikasi mobile
04/12/2023	Pembagian tugas perihal konsep dan pengembangan aplikasi mobile diagnosa kesehatan mental
12/12/2023	<ul style="list-style-type: none"> - Proses pembuatan <i>coding</i> program aplikasi diagnosa kesehatan mental serta menetukan fitur - fitur yang dibutuhkan untuk pembuatan aplikasi - Pembagian tugas terkait penyusunan laporan
14/12/2023	<ul style="list-style-type: none"> - Menjalankan Program aplikasi mobile yang sudah jadi 40% - Membahas mengenai metode yang akan digunakan dalam sistem pakar diagnosa kesehatan mental - Membahas mengenai rule based yang akan digunakan - Menyusun laporan tugas
24/12/2023	<ul style="list-style-type: none"> - Membahas dan menyusun ulang hasil dari Evaluasi yang diberikan oleh Pak Sawali mengenai rule based dan aturan metode forward chaining pada sistem pakar diagnosa kesehatan mental pada remaja - Review pembuatan dan hasil dari aplikasi yang dibuat - Pembagian tugas pembuatan poster, vidio animasi, ppt dan menyusun laporan akhir
05/01/2023	<ul style="list-style-type: none"> - Membuat konsep vidio animasi - Menyelesaikan laporan tugas

A. Lampiran

Notulen Rapat

1) Isi Notulen

Tanggal : 02 Desember 2023

Waktu : 1 jam

Tempat : Kantin Esa Unggul

Isi Notulen :

1. Diskusi tentang program aplikasi yang akan dibuat , mencari sumber-sumber referensi contoh pembuatan aplikasi diagnose Kesehatan mental remaja
2. Pembagian tugas perihal Tugas Project 1 mengenai konsep pembuatan aplikasi
3. Membahas Metode , UML pembuatan class diagram yang akan digunakan dalam sistem pakar diagnosis kesehatan mental serta membahasa konsep bentuk design aplikasi mobile

Tanggal : 14 Desember 2023

Waktu : 40 Menit

Tempat : Online Zoom

Isi Notulen :

1. Discuss dan review program aplikasi mobile diagnose kesehatan mental pada remaja , di review dari segi design, fitur, model apk dll
2. Masing-masing anggota kelompok memberi tanggapan review serta saran pada aplikasi , apakah ada yang perlu diperbaiki atau ada penambahan program
3. Pembagian tugas penyusuan laporan

Tanggal : 10 Januari 2024

Waktu : 1 Jam

Tempat : Online Zoom

Isi Notulen :

1. Discuss dan review kembali mengenai program aplikasi mobile diagnose kesehatan mental pada remaja yang sudah berjalan 90%

2. Membahas mengenai pembuatan video animasi yang akan dibuat serta membahas tugas akhir Project

Log Activity Anggota Kelompok

1) Anggota 1 – Ricky Salim

No	Tanggal	Durasi	Pekerjaan	Hasil
1	02/12/2023	50 Menit	Mempelajari soal Tugas Besar	Paham deskripsi permasalahan dari soal Tugas besar
2	04/12/2023	45 Menit	Mengimplementasikan Program	Memahami cara implementasi beserta relasi antar class
3	07/12/2023	20 Menit	Menggabungkan isi laporan Project 1 yang akan dikumpulkan	Laporan project 1 selesai
4	12/12/2023	1 Jam	Ikut serta dalam penyusuan laporan di awal project 1 & 2	Memahami laporan secara keseluruhan pada project 1 & project 2
			Memulai pengembangan modul autentikasi pengguna	Output bisa berjalan dengan sesuai
5	20/12/2023 Sampai 19/01/2024		1) Menulis unit tests untuk memastikan kualitas kode dan menemukan bugs pada tahap awal. 2) Melakukan code review bersama dengan anggota kelompok lain untuk memastikan kesesuaian kode dengan standar pengembangan yang telah ditetapkan.	<ul style="list-style-type: none"> ➤ Kode telah di-review, dioptimalkan, dan didokumentasikan dengan baik, siap untuk tahap produksi. ➤ Modul/Fitur yang dikembangkan telah terintegrasi dengan sukses dalam aplikasi utama tanpa masalah yang signifikan. ➤ Program berhasil

		<ul style="list-style-type: none"> 3) Optimalisasi kode dan refactoring untuk meningkatkan performa dan kebersihan kode. 4) Integrasi modul yang dikembangkan dengan bagian lain dari proyek. Pengujian integrasi dan penyelesaian konflik. 5) Dokumentasi kode dan pembuatan catatan rilis yang mencakup perubahan dan penambahan fitur. 6) Compile hasil program ke dalam bentuk Aplikasi 	dijalankan kedalam bentuk aplikasi
--	--	---	------------------------------------

2) Anggota 2 – Baby Anjel

No	Tanggal	Durasi	Pekerjaan	Hasil
1	02/12/2023	50 Menit	Mempelajari soal Tugas Besar	Paham deskripsi permasalahan dari soal Tugas besar
2	07/12/2023	40 Menit	Mencari sumber-sumber terkait tentang Functional Requirement dalam pembuatan Aplikasi	Memahami setiap fungsional requirement yang ada pada program
3	12/12/2023	1 Jam	Ikut serta dalam penyusuan laporan di awal project 1 & 2	Memahami laporan secara keseluruhan pada project 1 & project 2
4	22/12/2023	1 Jam	Memberikan info terkait point-point dalam pembahasan quisoner yang akan dimasukkan ke dalam program	Teredia 15 pertanyaan yang ditampilkan di dalam program

5	05/01- 12/01 2024	8 Hari	Membuat laporan yang diimplementasikan dalam bentuk PPT untuk presentasi	Laporan PPT berhasil diselesaikan
---	----------------------	--------	--	-----------------------------------

3) Anggota 3 – Sulistiyawati

No	Tanggal	Durasi	Pekerjaan	Hasil
1	02/12/2023	50 Menit	Mempelajari soal Tugas Besar	Paham deskripsi permasalahan dari soal Tugas besar
2	04/12/2023	1 Jam	Mencari sumber-sumber dari jurnal atau laporan terkait aplikasi yang akan dibuat dan menguraikan konsep overview project yang akan dihasilkan	Memahami konsep overview dari masalah serta project yang akan dihasilkan
3	12/12/2023	1 Jam	Ikut serta dalam penyusuan laporan di awal project 1 & 2	Memahami laporan secara keseluruhan pada project 1 & project 2
4	14/12/2023	1 Jam	Mencari info metode system pakar yang akan digunakan dalam pembuatan program aplikasi serta cara kerja metode tersebut	Menggunakan metode forward chaining
5	22/12/2023	20 Menit	Melakukan Test Modul Memahami prosedur pengujian modul	Memahami prosedur pengujian modul
6	08/01 – 12/01 2024	1 Minggu	Mereview dan menyelesaikan laporan tugas	Laporan Akhir Project

4) Anggota 4 - Adillah Saudah Shofa

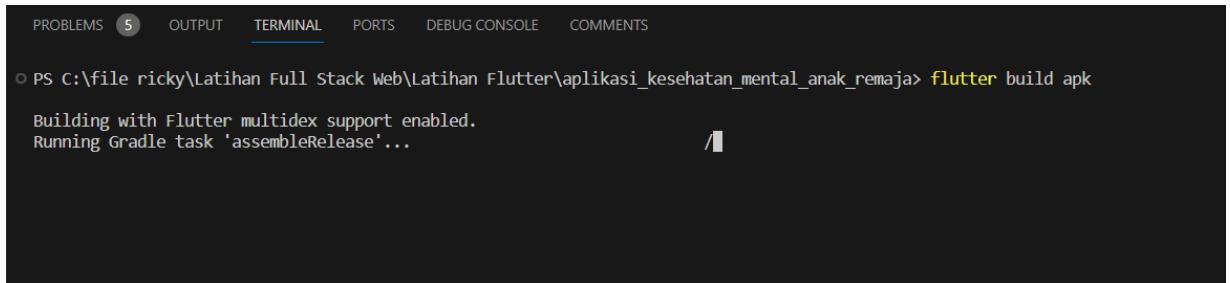
No	Tanggal	Durasi	Pekerjaan	Hasil
1	02/12/2023	50 Menit	Mempelajari soal Tugas Besar	Paham deskripsi permasalahan dari soal Tugas besar
2	07/12/2023	40 Menit	Membuat tabel activity log kelompok	Rincian log activity tiap anggota tersusun rapi dalam bentuk table
3	12/12/2023	1 Jam	Ikut serta dalam penyusuan laporan di awal project 1 & 2	Memahami laporan secara keseluruhan pada project 1 & project 2
4	06/01 – 11/01 2024	7 Hari	Membuat video animasi Aplikasi Diagnosa Kesehatan mental pada remaja	Pembuatan Vidio Animasi diselesaikan

5) Anggota 5 - Fikri Alam Arya Putra

No	Tanggal	Durasi	Pekerjaan	Hasil
1	02/12/2023	50 Menit	Mempelajari soal Tugas Besar	Paham deskripsi permasalahan dari soal Tugas besar
2	07/12/2023	40 Menit	Mencari kebutuhan non-fungsional dari project yang dibuat	Memahami kebutuhan non-fungsional pada program aplikasi yang dibuat
3	12/12/2023	1 Jam	Ikut serta dalam penyusuan laporan di awal project 1 & 2	Memahami laporan secara keseluruhan pada project 1 & project 2
4	25/12/2023	1 Jam	Mengembangkan service untuk kebutuhan mobile app	Menyelesaikan semua service yang dapat digunakan oleh mobile app
5	11/01/2024	1 Jam	Membuat Poster Aplikasi	Poster Aplikasi terselesaikan

B. Tutorial Cara Compile & Eksekusi Program

Jalankan flutter build apk



```
PROBLEMS 5 OUTPUT TERMINAL PORTS DEBUG CONSOLE COMMENTS

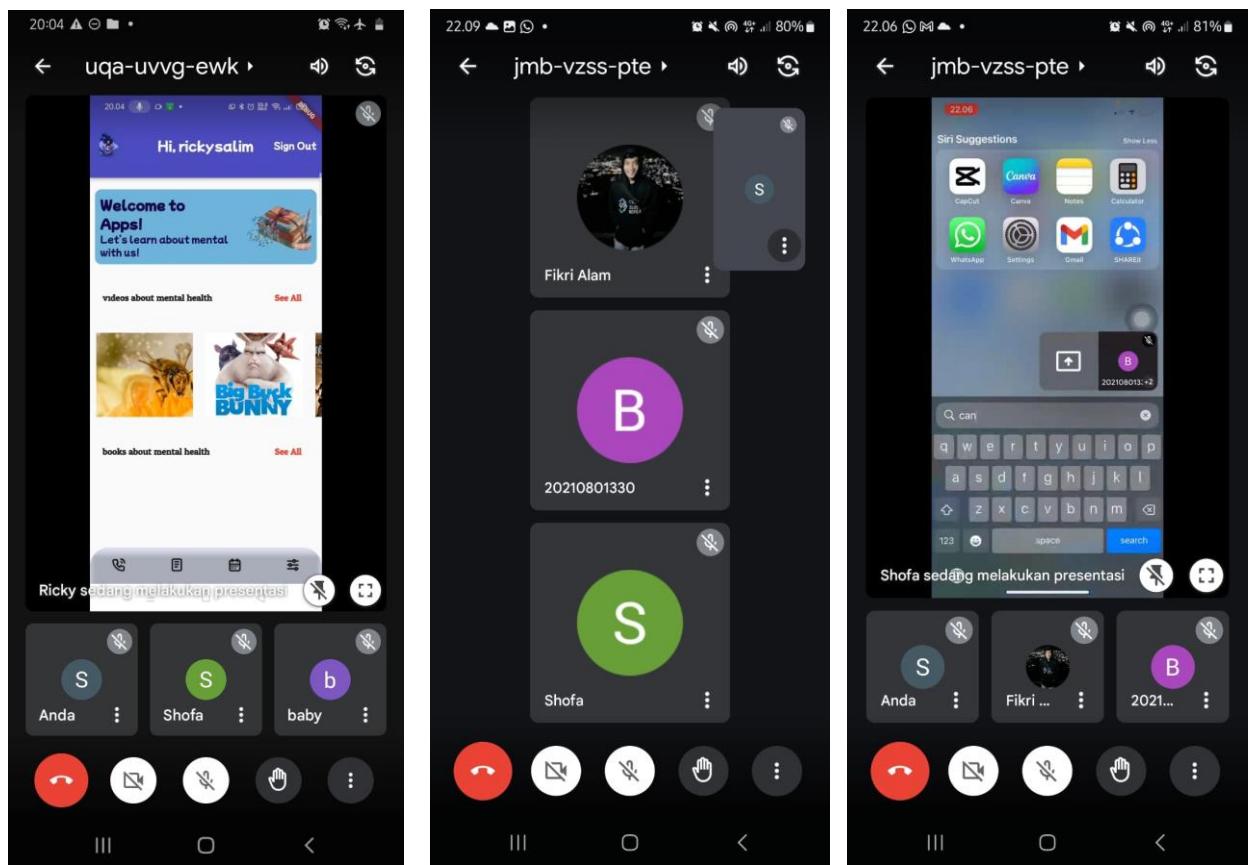
PS C:\file ricky\Latihan Full Stack Web\Latihan Flutter\aplikasi_kesehatan_mental_anak_remaja> flutter build apk
Building with Flutter multidex support enabled.
Running Gradle task 'assembleRelease'...
```

Setelah selesai masuk ke folder build\app\outputs\flutter-apk\app-release.apk



```
Running Gradle task 'assembleRelease'... 162.7s
✓ Built build\app\outputs\flutter-apk\app-release.apk (32.2MB).
PS C:\file ricky\Latihan Full Stack Web\Latihan Flutter\aplikasi_kesehatan_mental_anak_remaja>
```

C. Dokumentasi Koordinasi



D. Penyertaan Model Analisis PIECES



E. Requirement Systems

1) Kebutuhan Fungsional

Functional Requirement (Kebutuhan Fungsional) adalah spesifikasi yang mendeskripsikan perilaku yang diinginkan atau fungsi yang harus dimiliki oleh suatu sistem perangkat lunak atau aplikasi. Kebutuhan fungsional merinci secara rinci apa yang diharapkan sistem lakukan, bagaimana berbagai input harus ditangani, dan apa output yang dihasilkan.

No.	Kebutuhan	Deskripsi	Detail Tambahan
1	Autentikasi Pengguna	Pengguna harus dapat membuat akun dan masuk dengan aman.	Implementasikan metode otentikasi yang aman seperti verifikasi email atau otentikasi dua faktor.
2	Manajemen Profil	Pengguna dapat membuat, mengedit, dan menghapus profil mereka.	Sertakan fitur seperti gambar profil, usia, jenis kelamin, dan informasi relevan lainnya.
3	Kuisoner Awal	Aplikasi menyediakan kuesioner awal untuk	Kuesioner ini menggunakan pertanyaan

		mengidentifikasi gejala dan masalah kesehatan mental yang mungkin dialami pengguna.	pilihan ganda atau skala Likert.
4	Diagnosis Otomatis	Berdasarkan jawaban dari kuesioner, aplikasi memberikan diagnosis awal atau saran mengenai kondisi kesehatan mental yang mungkin dialami pengguna.	Aplikasi menyediakan informasi dan sumber tentang kondisi yang didiagnosa.
4	Dukungan Krisis	Sediakan informasi kontak darurat dan akses ke jalur bantuan krisis.	Sertakan fitur dukungan segera, seperti panggilan darurat satu kali klik atau dukungan obrolan.
5	Pustaka Sumber Daya	Repositori artikel, video, dan sumber daya terkait kesehatan mental.	Kategorikan sumber daya berdasarkan topik seperti stres, kecemasan, depresi, dll.
6	Privasi dan Keamanan	Implementasikan langkah-langkah kokoh untuk melindungi data pengguna dan menjaga kerahasiaan.	Patuhi peraturan perlindungan data dan pastikan penyimpanan data yang aman.
7	Umpulan dan Penilaian	Biarkan pengguna memberikan umpan balik dan menilai aplikasi.	Gunakan umpan balik untuk meningkatkan fitur dan kegunaan aplikasi.
8	Fitur Aksesibilitas	Pastikan aplikasi dapat diakses oleh pengguna dengan disabilitas.	Sertakan fitur seperti perintah suara, pembaca layar, dan mode kontras tinggi.

2) Kebutuhan Non-Fungsional

- Antarmuka Pengguna yang Mudah: Antarmuka yang intuitif dan mudah dinavigasi agar remaja dapat menggunakan aplikasi tanpa kesulitan.
- Kuesioner Penilaian: Modul yang berisi pertanyaan dirancang untuk mengidentifikasi gejala umum gangguan kesehatan mental pada remaja.
- Modul Edukasi: Memberikan informasi tentang berbagai aspek kesehatan mental, termasuk mengenali gejala, pentingnya mencari bantuan, dan strategi pengelolaan.
- Sistem Rekomendasi: Berdasarkan tanggapan atas kuesioner, aplikasi dapat memberikan rekomendasi tindakan awal, seperti menghubungi profesional kesehatan mental.
- Umpam Balik dan Evaluasi: Fitur untuk mengumpulkan umpan balik pengguna tentang aplikasi untuk peningkatan dan pembaruan berkelanjutan.

F. Metode Khusus Algoritma Untuk (SPK / Sistem Pakar / Aplikasi Lainnya)

1) Forward Chaining:

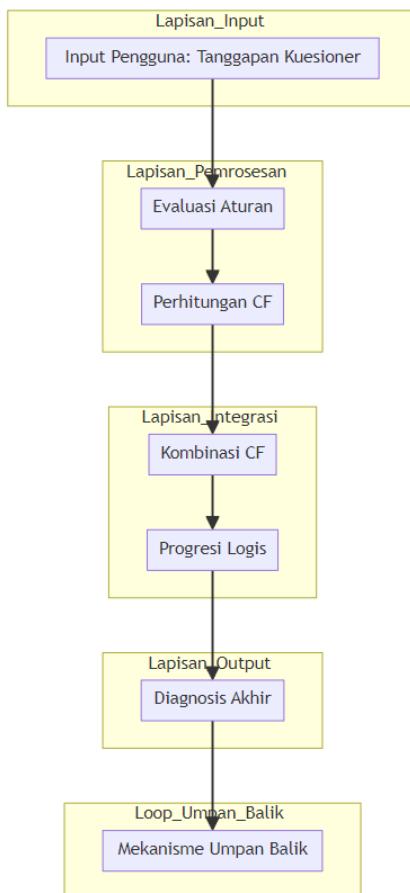
- Berbasis Aturan: Forward Chaining bekerja dengan aturan IF-THEN. Misalnya, "IF pengguna merasa sedih selama dua minggu, THEN mungkin mengalami depresi."
- Proses Deduktif: Aplikasi secara berurutan memeriksa setiap aturan, mencocokkan kondisi IF dengan informasi dari pengguna, dan mengarahkan ke kesimpulan logis.
- Langkah demi Langkah: Setiap kali aturan dipenuhi, aplikasi membuat inferensi dan melanjutkan ke aturan berikutnya, secara bertahap membangun diagnosis.

2) Certainty Factor (CF):

- Skala Kepercayaan: CF menilai tingkat kepercayaan dari setiap gejala, dengan skala dari -1 (pasti tidak) hingga 1 (pasti ya).
- Kombinasi Nilai: Nilai CF dari pengguna (berdasarkan pengalaman gejala) dikombinasikan dengan nilai CF dari pakar (berdasarkan pengetahuan medis).

- Formula CF: Misalnya, jika CF1 (paket) adalah 0.8 dan CF2 (pengguna) adalah 0.7, maka CF kombinasi dihitung dengan $CF_{kombinasi} = 0.8 + 0.7 \times (1 - 0.8)$

3) Alur Sistem Pakar



- ❖ Input Pengguna: Pengguna memasukkan data gejala melalui kuesioner dalam aplikasi.
- ❖ Identifikasi Gejala Awal: Sistem mengidentifikasi gejala awal berdasarkan input pengguna.
- ❖ Evaluasi Aturan Forward Chaining: Sistem mengevaluasi aturan-aturan berdasarkan gejala yang telah diidentifikasi. Aturan ini mengikuti format IF-THEN, dimana 'IF' merupakan kondisi gejala dan 'THEN' merupakan kesimpulan sementara.
- ❖ Perhitungan Certainty Factor (CF): Untuk setiap gejala, CF dihitung berdasarkan input pengguna dan penilaian pakar. CF ini merepresentasikan tingkat kepercayaan terhadap keterkaitan gejala dengan kondisi kesehatan mental tertentu.
- ❖ Aggregasi CF: CF dari berbagai gejala digabungkan untuk mendapatkan gambaran keseluruhan yang lebih akurat.
- ❖ Penarikan Kesimpulan: Berdasarkan CF yang teragregasi, sistem menarik kesimpulan atau diagnosis.
- ❖ Hasil dan Rekomendasi: Sistem menyajikan hasil diagnosa kepada pengguna, beserta rekomendasi tindak lanjut jika diperlukan.
- ❖ Umpan Balik dan Penyesuaian: Sistem memungkinkan pengguna untuk memberikan umpan balik, yang dapat digunakan untuk meningkatkan akurasi dan efektivitas sistem di masa depan.

4) Database dan Logic Diagram

Database: Berisi daftar gejala dengan CF terkait dan aturan yang digunakan dalam Forward Chaining.

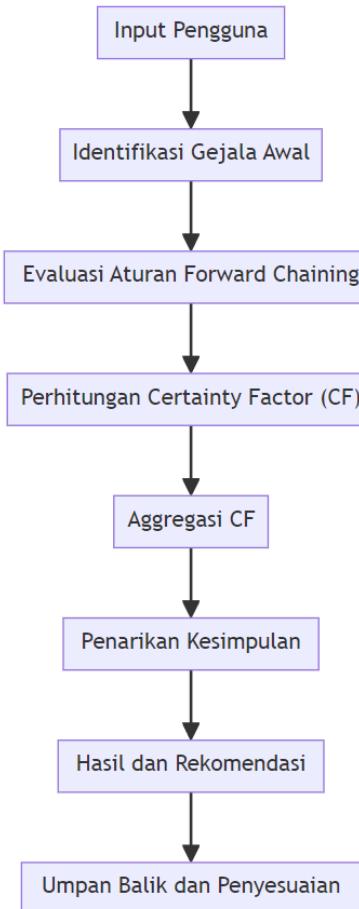
Certainty Faktor Gejala dari Pakar

No	Kode	Gejala	CF
1	G01	Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah.	0.7
2	G02	Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas.	0.8
3	G03	Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih.	0.6
4	G04	Pola makan yang tidak sehat, obsesi dengan berat badan.	0.5
5	G05	Kesulitan tidur atau tidur berlebihan.	0.6
6	G06	Kelelahan berkelanjutan, kurangnya motivasi.	0.7
7	G07	Perasaan tidak berharga atau bersalah tanpa alasan yang jelas.	0.75
8	G08	Kesulitan untuk fokus, mudah terdistraksi.	0.5
9	G09	Menghindari interaksi sosial, menyendiri.	0.65
10	G10	Pikiran untuk menyakiti diri sendiri, perilaku merusak diri.	0.85
11	G11	Penurunan atau kenaikan berat badan yang signifikan tanpa alasan yang jelas.	0.5
12	G12	Reaksi emosional yang berlebihan atau tidak sesuai.	0.55
13	G13	Tidak dapat melihat hal yang positif dari suatu kegiatan kejadian	0.6
14	G14	Merasa putus asa dan hilang harapan	0.8
15	G15	Merasa tidak ada hal yang dapat diharapkan di masa depan	0.8
16	G16	Merasa sulit untuk meningkatkan inisiatif dalam melakukan sesuatu	0.6

Aturan Forward Chaining

Gangguan	Aturan (Rules)
KM01	IF presentase \leq 20% THEN Diagnosis = Normal (Tidak ada Gangguan)
KM02	IF presentase $>$ 20% dan \leq 40% THEN Diagnosis = Depresi Ringan
KM03	IF presentase $>$ 40% dan \leq 60% THEN Diagnosis = Depresi Sedang
KM04	IF presentase $>$ 60% dan \leq 80% THEN Diagnosis = Depresi Berat Tanpa Gangguan Fungsional
KM05	IF presentase $>$ 80% THEN Diagnosis = Depresi Berat dengan Gangguan Fungsional

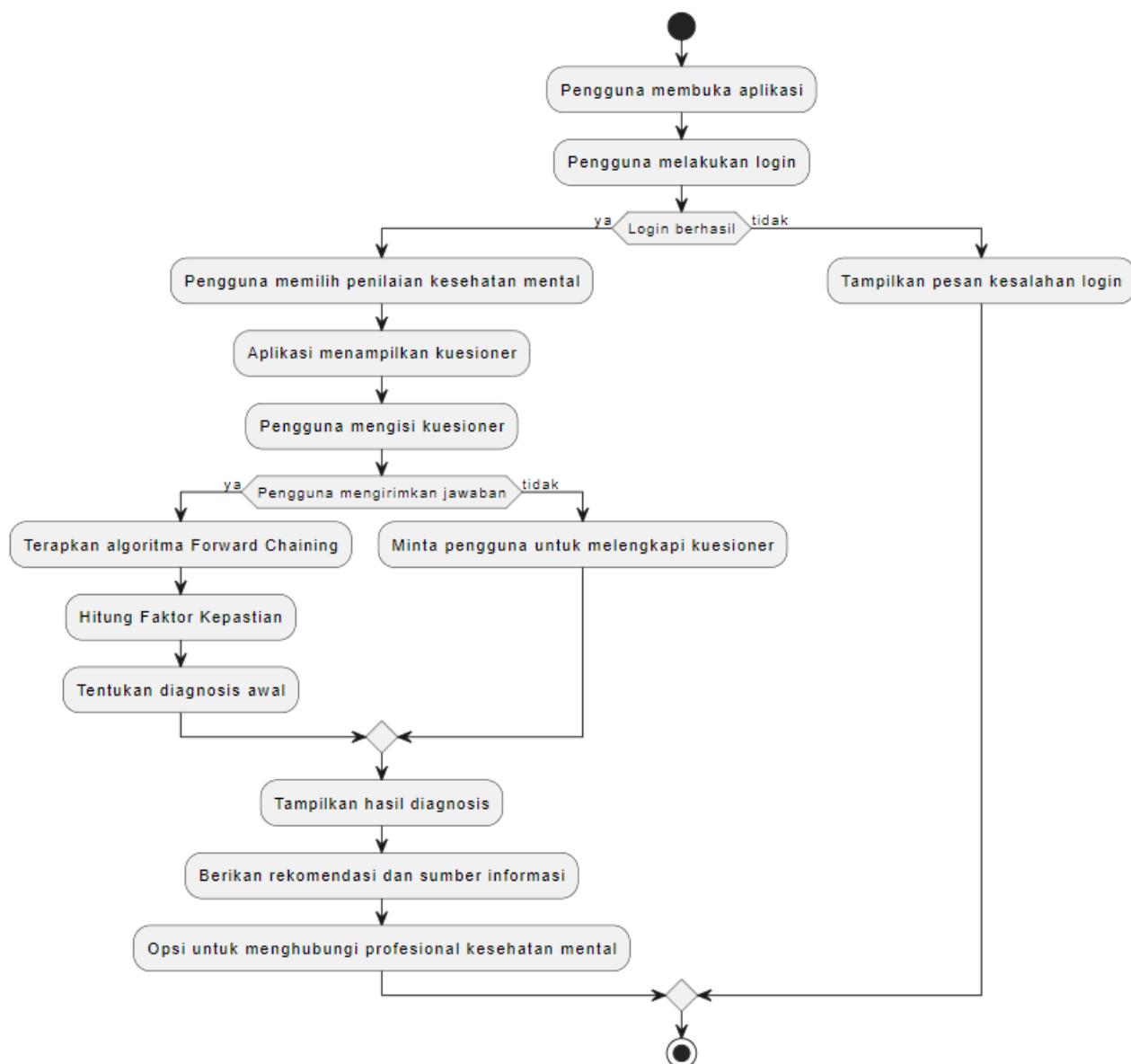
Logic Diagram: Menggambarkan proses dari pengambilan input, penerapan aturan, perhitungan CF, hingga output diagnosis.



1. Lapisan Input: Diagram dimulai dengan input pengguna, yaitu tanggapan terhadap kuesioner penilaian kesehatan mental dalam aplikasi.
2. Lapisan Pemrosesan:
 - Evaluasi Aturan: Di sini, setiap input pengguna (gejala) dicocokkan dengan bagian IF dari aturan IF-THEN (misalnya, "Jika merasa sedih lebih dari dua minggu, maka pertimbangkan depresi").
 - Perhitungan CF: Untuk setiap aturan yang cocok, hitung Faktor Kepastian berdasarkan CF pakar untuk gejala tersebut dan CF yang dilaporkan pengguna.
3. Lapisan Integrasi:
 - Kombinasi CF: Gabungkan CF dari berbagai aturan menggunakan rumus kombinasi CF.
 - Progresi Logis: Terapkan logika Forward Chaining untuk melanjutkan ke aturan relevan berikutnya berdasarkan hasil evaluasi aturan sebelumnya.

4. Lapisan Output: Diagnosis akhir atau status kesehatan mental diturunkan berdasarkan CF teragregasi dan kesimpulan yang dicapai melalui proses Forward Chaining.
5. Loop Umpan Balik: Opsional, diagram dapat menyertakan mekanisme umpan balik di mana interaksi pengguna dengan diagnosis dapat digunakan untuk penyempurnaan sistem.

G. Desain Perancangan Sistem



H. Desain UI



GET STARTED





Sign In

Need to create an account? [Sign Up](#)

Sign Up

Already have an account? [Sign In](#)

Video Learning



Pentingnya Menjaga Kesehatan



DEPRESI Penyakit yang NGUDA sembarangan, tapi masih sering DIABAIKAN

Diagnose





Relaxing Music



Good Night



Once In Paris

Breathing Exercise



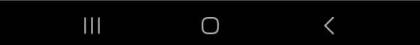


Diagnose



Help





Rasa sedih ngga butuh perkenalan. Kita semua pasti pernah ngerasain sedih; saat kita gagal ngelakuin sesuatu; pas kita ngerasa kesepian; atau ketika kita kehilangan orang yang tersayang. Tapi kadang, ada sesuatu yang sebenarnya lebih dari sekadar perasaan sedih. Sesuatu yang seringnya kita abaikan. Enjoy the video! Omong-omong, video ini hasil kolaborasi Kok Bisa bersama Johnson & Johnson! Untuk memanusiakan depresi dan ngegambarin dampaknya ke kita-kita, video ini menampilkan Alex, karakter non-biner yang diciptakan oleh J&J yang sedang mengalami depresi, bersama dengan tiga blue blobs, yang masing-masing ngegambarin tiga tingkatan yang berbeda dari depresi: Ringan, sedang dan berat. Kita ngerasa pesan mengenai kesehatan mental itu super penting dan perlu untuk banyak dibicarakan—terlebih soal depresi. Makanya, buat kalian yang ngerasa lagi di kondisi depresi, atau ngalamin masalah kesehatan mental yang lain, segera cari bantuan medis profesional, ya! Tenang aja, gak usah malu. Karena ibarat penyakit yang lain, penyakit ini bukan tanda kalo kita lemah — melainkan emang cuma butuh diobatin.

Once In Paris

00:00:08 00:02:04

Profile

Diagnose History

Consultation

Sign Out

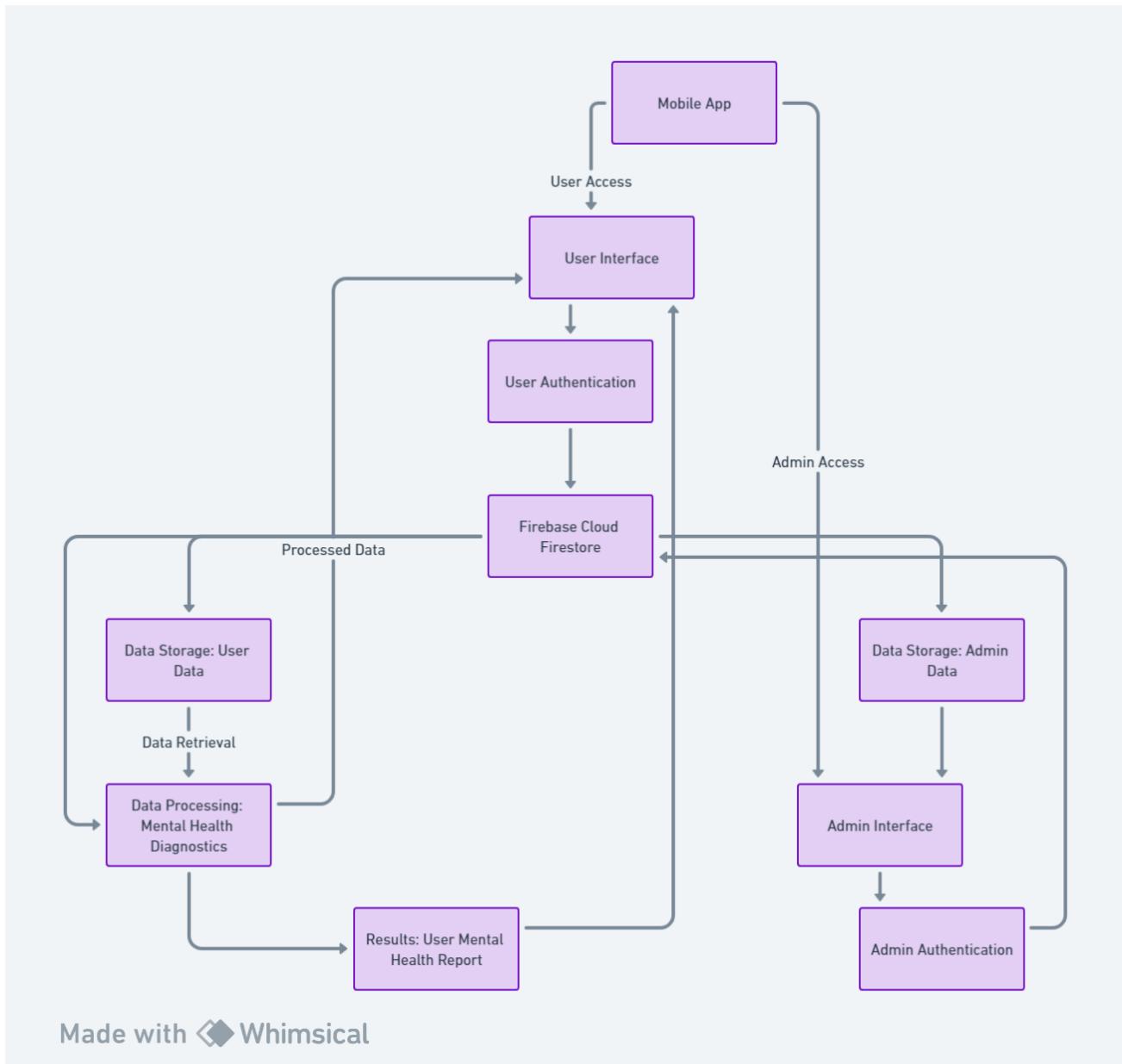
05.41 66%

Hi rickysalim2

Music Video

Question Answer Rule Based

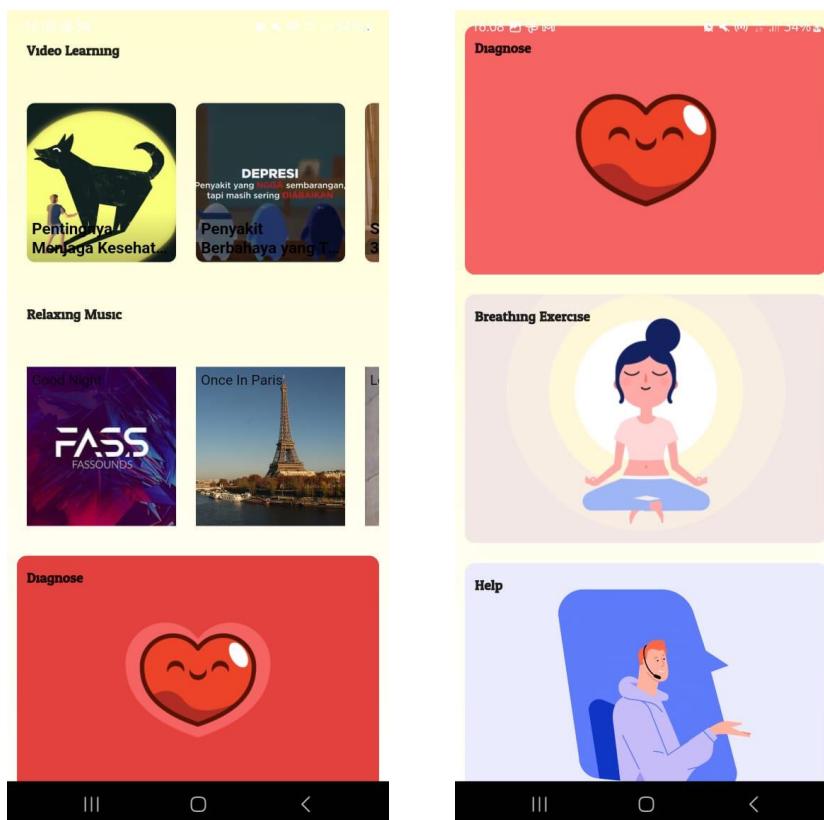
I. Arsitektur Aplikasi



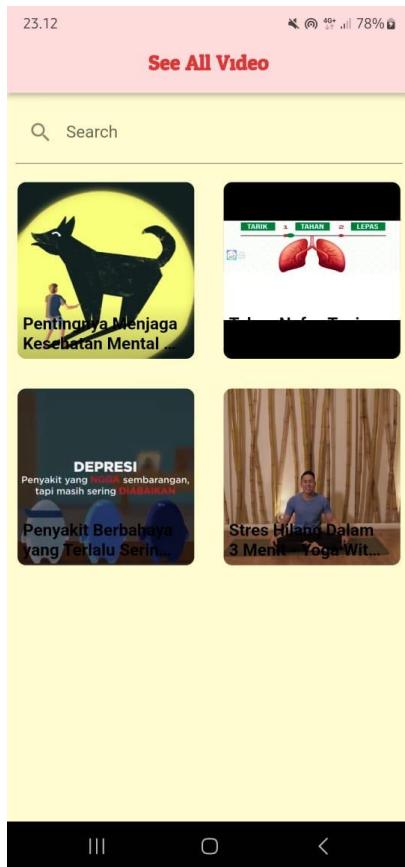
J. Flow Tampilan Akhir Program

<p>Login</p>  <div style="margin-top: 10px;"> <input type="text" value="Email"/> <input type="password" value="Password"/> </div> <div style="background-color: #002B36; color: white; padding: 5px; text-align: center; border-radius: 5px; width: fit-content; margin: auto;"> Sign In </div> <div style="text-align: center; margin-top: 10px;"> <small>Need to create an account? Sign Up</small> </div> <div style="background-color: black; height: 20px; width: 100%; position: relative;"> III O < </div>	<p>Register</p>  <div style="margin-top: 10px;"> <input type="text" value="Username"/> <input type="text" value="Email"/> <input type="password" value="Password"/> <input type="password" value="Confirm Password"/> </div> <div style="background-color: #002B36; color: white; padding: 5px; text-align: center; border-radius: 5px; width: fit-content; margin: auto;"> Sign Up </div> <div style="text-align: center; margin-top: 10px;"> <small>Already have an account? Sign In</small> </div> <div style="background-color: black; height: 20px; width: 100%; position: relative;"> III O < </div>
---	---

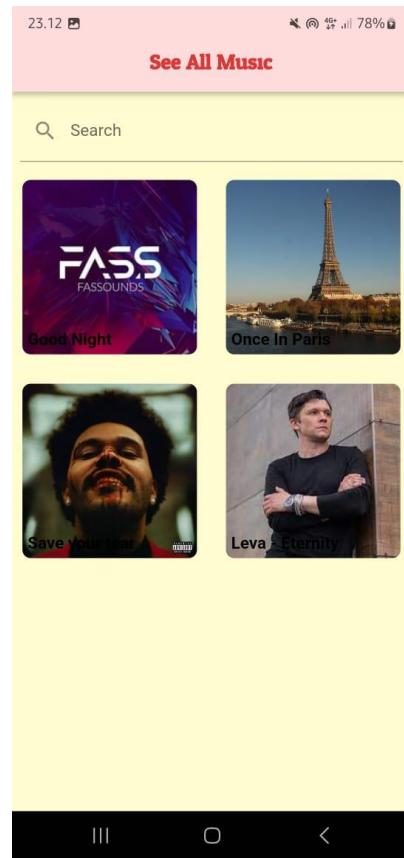
Main Menu (Tampilan Utama Beranda)



Tampilan HomePage Vidio



Tampilan HomePage Music



Tampilan Quisioner

16.52 51%

Pikiran untuk menyakiti diri sendiri, perilaku merusak diri

sangat sering

lumayan sering

kadang-kadang

tidak pernah

[Next](#)

16.53 51%

merasa putus asa dan hilang harapan

sangat sering

lumayan sering

kadang-kadang

tidak pernah

[Prev](#) [Next](#)

16.55 ☀️ 50%

reaksi emosional yang berlebihan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Submit**

16.54 ☀️ 50%

merasa tidak ada hal yang diharapkan di masa depan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.55 ☀️ 50%

Perubahan mood yang cepat dan ekstrem, dari sangat bahagia ke sangat sedih.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.53 ☀️ 51%

kelelahan tidur atau tidur berlebihan

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.55 ☀️ 50%

Perasaan sedih yang berkepanjangan, kehilangan minat atau kesenangan dalam aktivitas.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.53 ☀️ 51%

Kekhawatiran berlebih, kesulitan mengontrol kecemasan, perasaan gelisah.

sangat sering
 lumayan sering
 kadang-kadang
 tidak pernah

Prev **Next**

16.54 50%

pola makan tidak sehat, obsesi dengan berat badan

sering sekali

lumayan sering

kadang-kadang

tidak pernah

Prev **Next**

16.55 50%

kelelahan berkelanjutan, kurang motivasi

sangat sering

lumayan sering

kadang-kadang

tidak pernah

Prev **Next**

16.53 51%

penurunan atau kenaikan berat badan secara signifikan tanpa alasan jelas.

sangat sering

lumayan sering

kadang-kadang

tidak pernah

Prev **Next**

16.54 50%

merasa sulit meningkatkan inisiatif dalam melakukan sesuatu

sangat sering

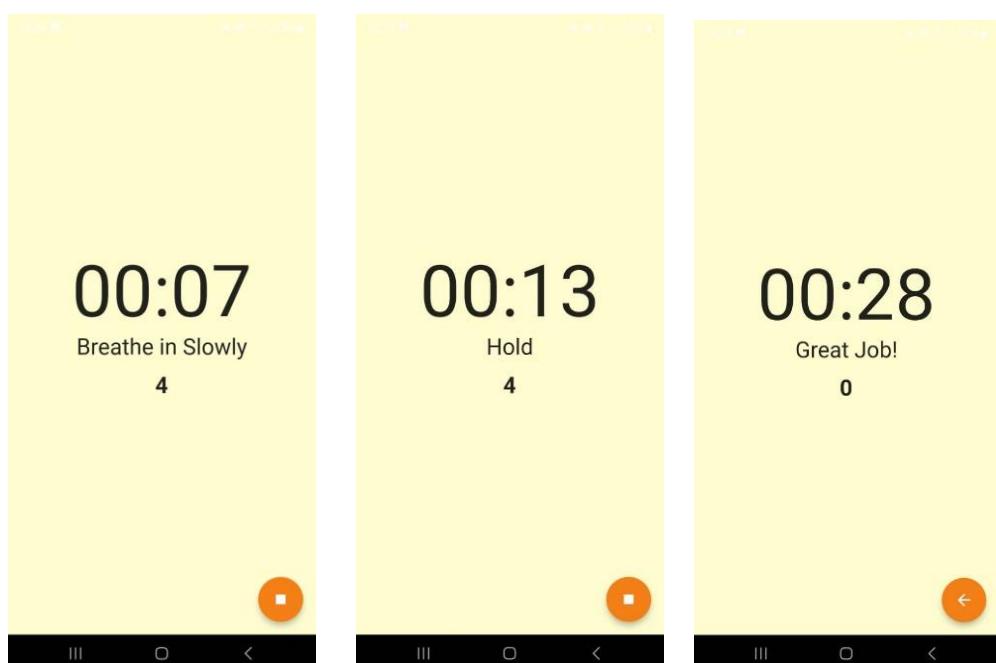
lumayan sering

kadang-kadang

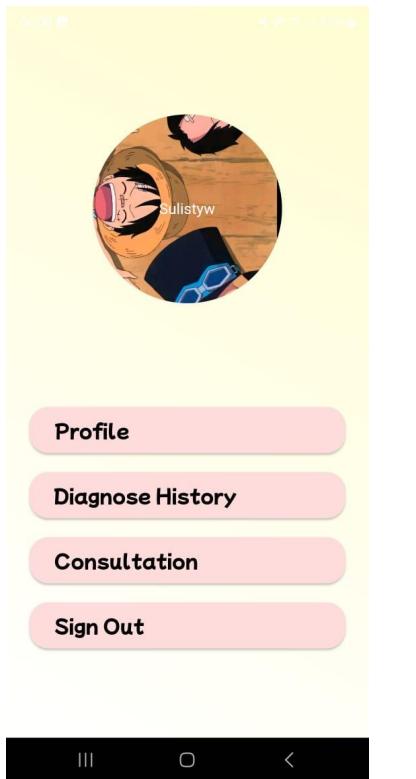
tidak pernah

Prev **Next**

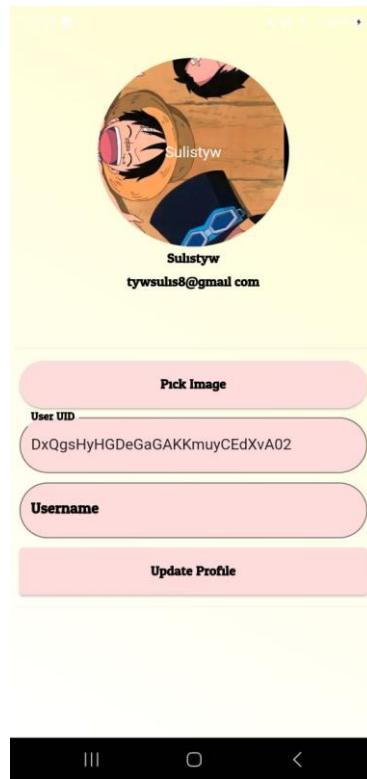
Tampilan Breathing Exercise



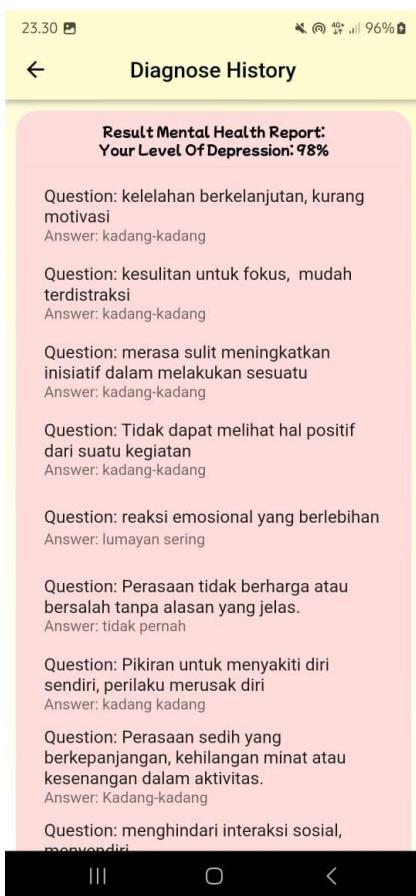
Tampilan Menu “Setting”



Tampilan Profile User



Tampilan Diagnose History



Tampilan Consultation



K. Programming Source Code dan Database Design

Bagian Controller

audio_controller.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/music_repository.d
art';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/music_screen.dart'
;
import 'package:audioplayers/audioplayers.dart';
import 'package:get/get.dart';

class MusicController extends GetxController {
    static MusicController get instance => Get.find();

    late AudioPlayer audioPlayer;

    RxBool.isPlaying = RxBool(false);

    Rx<Duration> duration = Rx<Duration>(const Duration(seconds: 0));
    Rx<Duration> position = Rx<Duration>(const Duration(seconds: 0));

    RxString urlAudio = RxString("");

    void setAudio(Music music) {
        Get.to(MusicScreen(music));
        urlAudio.value = music.musicUrl!;
        update();
    }
}
```

```

void slider(double value) {
    audioPlayer.seek(Duration(seconds: value.toInt()));
    audioPlayer.resume();
}

String formatTime(int seconds) {
    return '${(Duration(seconds: seconds))}'.split('.')[0].padLeft(8, '0');
}

Future<void> onClickEvent() async {
   .isPlaying.value
    ? await audioPlayer.pause()
    : await audioPlayer.play(UrlSource(urlAudio.value));
}

@Override
void onInit() {
    super.onInit();
    audioPlayer = AudioPlayer();
    audioPlayer.onPlayerStateChanged.listen((event) {
        if (event == PlayerState.playing) {
           .isPlaying.value = true;
        } else if (event == PlayerState.paused ||
                    event == PlayerState.stopped ||
                    event == PlayerState.completed) {
           .isPlaying.value = false;
        }
        update();
    });
}

audioPlayer.onDurationChanged.listen((event) {
    duration.value = event;
}

```

```

        update();
    });

audioPlayer.onPositionChanged.listen((event) {
    position.value = event;
    update();
});
}

void disposeAudioPlayer() {
    isPlaying.value = false;
    duration.value = Duration.zero;
    position.value = Duration.zero;
    urlAudio.value = "";
    musicName.value = "";
    audioPlayer.dispose();
}
}

final musicRepositoryController = Get.put(MusicRepositoryController());

RxString musicName = RxString("");

void clearSearch() {
    musicName.value = "";
    update();
}

void setSearch(String value) {
    musicName.value = value;
    update();
}

```

```
Stream<List<Music>> searchMusic() =>
    musicRepositoryController.searchMusic(musicName.value);

Stream<List<Music>> getAllMusic() => musicRepositoryController.getAllMusic();
}
```

auth_controller.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/authentication_reposi
tory_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/utils/exceptions/signin_exception.da
rt';
import
'package:aplikasi_kesehatan_mental_anak_remaja/utils/exceptions/signup_exception.da
rt';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class AuthController extends GetxController {
    static AuthController get instance => Get.find();

    final authRepoController = Get.put(AuthenticationRepositoryController());

    final username = TextEditingController();
    final email = TextEditingController();
    final password = TextEditingController();
    final confirmPassword = TextEditingController();

    void resetText() {
```

```
email.clear();
username.clear();
password.clear();
confirmPassword.clear();
}
```

```
Future<void> register() async {
try{
await authRepoController.createUserWithEmailAndPassword(email.text,
password.text, username.text);
resetText();
} on SignUpWithEmailAndPasswordFailureException catch (e) {
Get.snackbar(e.code, e.message);
}
}
```

```
Future<void> logout() async {
try{
await authRepoController.signOut();
} catch(e) {
Get.snackbar('Error', e.toString());
}
}
```

```
Future<void> login() async {
try{
await authRepoController.signInWithEmailAndPassword(email.text,
password.text);
resetText();
} on SignInWithEmailAndPasswordFailureException catch (e) {
Get.snackbar(e.code, e.message);
}
}
```

}

breathing_exercise_controller.dart

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

```
enum SessionState {
```

```
    initial,
    starting,
    holdBreathIn,
    holdBreathOut,
    breathingIn,
    breathingOut,
    ended,
    invalid
```

}

```
class BreatheSession {
```

```
    late int inBreaths;
    late int outBreaths;
    late double sessionLengthSeconds;
}
```

```
class TimerModel {
```

```
    final RxList<ValueChanged<ElapsedTime>> timerListeners =
        RxList(<ValueChanged<ElapsedTime>>[]);
    final Rx<TextStyle> textStyle =
        Rx(const TextStyle(fontSize: 90.0, fontFamily: "Bebas Neue"));
    final Rx<Stopwatch> stopwatch = Rx(Stopwatch());
    final RxInt timerMillisecondsRefreshRate = RxInt(30);
```

```
}
```

```
class ElapsedTime {
```

```
    final int seconds;
```

```
    final int minutes;
```

```
ElapsedTime({
```

```
    required this.seconds,
```

```
    required this.minutes,
```

```
});
```

```
}
```

```
class BreathingExerciseController extends GetxController {
```

```
    static BreathingExerciseController get instance => Get.find();
```

```
Rx<TimerModel> dependencies = Rx<TimerModel>(TimerModel());
```

```
Rx<Timer?> timer = Rx<Timer?>(null);
```

```
late RxInt milliseconds = RxInt(0);
```

```
RxInt minutes = RxInt(0);
```

```
RxInt seconds = RxInt(0);
```

```
Rx<SessionState> sessionState = Rx<SessionState>(SessionState.initial);
```

```
RxInt countDown = RxInt(0);
```

```
Rx<Timer?> countDownTimer = Rx<Timer?>(null);
```

```
Rx<Duration> oneSec = Rx<Duration>(const Duration(seconds: 1));
```

```
void onTick(ElapsedTime elapsed) {
```

```
    if (elapsed.minutes != minutes.value || elapsed.seconds != seconds.value) {
```

```
        minutes.value = elapsed.minutes;
```

```
        seconds.value = elapsed.seconds;
```

```
    }
```

```
}
```

```
@override
```

```

void onInit() {
    timer.value = Timer.periodic(
        Duration(milliseconds: dependencies.value.timerMillisecondsRefreshRate.value),
        callback);
    dependencies.value.timerListeners.add(onTick);
    super.onInit();
}

void resetAll() {
    dependencies.value.stopwatch.value.reset();
    milliseconds.value = 0;
    minutes.value = 0;
    seconds.value = 0;
    sessionState.value = SessionState.initial;
    countDown.value = 0;
    countDownTimer.value?.cancel();
    countDownTimer.value = null;
    Get.back();
}

void callback(Timer timer) {
    if (milliseconds.value != dependencies.value.stopwatch.value.elapsedMilliseconds) {
        milliseconds.value = dependencies.value.stopwatch.value.elapsedMilliseconds;
        final RxInt hundreds = RxInt((milliseconds.value / 10).truncate());
        final RxInt seconds = RxInt((hundreds.value / 100).truncate());
        final RxInt minutes = RxInt((seconds.value / 60).truncate());
        final ElapsedTime elapsedTime =
            ElapsedTime(seconds: seconds.value, minutes: minutes.value);
        for (final listener in dependencies.value.timerListeners) {
            listener(elapsedTime);
        }
    }
}
    
```

```

void start() {
    dependencies.value.stopwatch.value.start();
    beginExcerciseRoutine();
}

void stop() {
    dependencies.value.stopwatch.value.stop();
    sessionState.value = SessionState.ended;
    countDownTimer.value!.cancel();
    countDown.value = 0;
}

void beginExcerciseRoutine() {
    sessionState.value = nextState(sessionState.value);
    countDown.value = 5;
    countDownTimer.value = Timer.periodic(oneSec.value, (timer) {
        countDown.value--;
        if (countDown.value < 0) {
            countDown.value = 5;
            sessionState.value = nextState(sessionState.value);
        }
    });
}

SessionState nextState(SessionState state) {
    SessionState next;
    switch (state) {
        case SessionState.initial:
            next = SessionState.starting;
            break;
        case SessionState.starting:
            next = SessionState.breathingIn;
    }
}

```

```

        break;

    case SessionState.breathingIn:
        next = SessionState.holdBreathIn;
        break;

    case SessionState.breathingOut:
        next = SessionState.holdBreathOut;
        break;

    case SessionState.holdBreathIn:
        next = SessionState.breathingOut;
        break;

    case SessionState.holdBreathOut:
        next = SessionState.breathingIn;
        break;

    case SessionState.ended:
        next = SessionState.ended;
        break;

    default:
        next = SessionState.invalid;
        break;
    }

    return next;
}

```

```

String instructionText(SessionState state) {
    RxString text = RxString("");
    switch (state) {
        case SessionState.initial:
            text.value = "Press Play to Begin";
            break;
        case SessionState.starting:
            text.value = "Get Ready... ";
            break;
        case SessionState.breathingIn:

```

```

text.value = "Breathe in Slowly";
break;

case SessionState.breathingOut:
    text.value = "Breathe out Slowly";
    break;

case SessionState.holdBreathIn:
case SessionState.holdBreathOut:
    text.value = "Hold";
    break;

case SessionState.ended:
    text.value = "Great Job!";
    break;

default:
    text.value = "INVALID STATE";
    break;
}

return text.value;
}
}

```

crisis_support_controller.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/crisis_support_controller_repository.dart';

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/crisis_support.dart';
import 'package:get/get.dart';

class CrisisSupportController extends GetxController {

```

```

static CrisisSupportController getInstance => Get.find();

final crisisSupportRepositoryController =
Get.put(CrisisSupportRepositoryController());

Stream<List<CrisisSupport>> getAllCrisisSupport() =>
crisisSupportRepositoryController.getAllCrisisSupport();

}

```

diagnose_controller.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/diagnose_repositor
y_controller.dart';

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class DiagnoseController extends GetxController {
    static DiagnoseController getInstance => Get.find();

    RxMap<String, dynamic> mapSelectedOptions = RxMap<String, dynamic>({ });
    RxBool showSubmitButton = RxBool(false);
    RxList<String> questions = RxList<String>();
    RxDouble resultTest = RxDouble(0.0);

    void setShowSubmitButton(bool val) {
        showSubmitButton.value = val;
        update();
    }

    void setQuestions(List<String> questionList) {

```

```
questions.addAll(questionList);
}

Widget isRadioButtonNull(String question) {
    if (mapSelectedOptions[question] == null) {
        return const Align(
            alignment: Alignment.centerLeft,
            child: Padding(
                padding: EdgeInsets.all(10),
                child: Text('Tidak boleh kosong',
                    textAlign: TextAlign.left,
                    style: TextStyle(
                        fontFamily: 'MochiyPopOne',
                        fontSize: 10,
                        fontWeight: FontWeight.w400,
                        color: Colors.red,
                    ))));
    }
    return const Text(");
}

bool allOptionsSelected() {
    for (String question in questions) {
        if (mapSelectedOptions[question] == null) {
            return false;
        }
    }
    return true;
}

void setSelectionOptions(String question, String value) {
    mapSelectedOptions[question] = value;
    update();
}
```

```
}
```

```

void abortAllSelectionOptions() {
    questions.clear();
    mapSelectedOptions.clear();
    resultTest.value = 0.0;
}
```

```
final diagnoseRepositoryController = Get.put(DiagnoseRepositoryController());
```

```

Future<void> calculateDiagnose() async {
    double result = await
diagnoseRepositoryController.calculateDiagnose(mapSelectedOptions);
    int nilaiPersentase = (result * 100).toInt();
    resultTest.value = nilaiPersentase.toDouble();
    update();
}
```

```

Stream<List<Diagnose>> getAllDiagnose() {
    return diagnoseRepositoryController.getAllDiagnose();
}
}
```

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/diagnose_repositor
y_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
```

```

class DiagnoseController extends GetxController {
    static DiagnoseController get instance => Get.find();
```

```

RxMap<String, dynamic> mapSelectedOptions = RxMap<String, dynamic>({ });
RxBool showSubmitButton = RxBool(false);
RxList<String> questions = RxList<String>();
RxDouble resultTest = RxDouble(0.0);

void setShowSubmitButton(bool val) {
    showSubmitButton.value = val;
    update();
}

void setQuestions(List<String> questionList) {
    questions.addAll(questionList);
}

Widget isRadioButtonNull(String question) {
    if (mapSelectedOptions[question] == null) {
        return const Align(
            alignment: Alignment.centerLeft,
            child: Padding(
                padding: EdgeInsets.all(10),
                child: Text('Tidak boleh kosong',
                    textAlign: TextAlign.left,
                    style: TextStyle(
                        fontFamily: 'MochiyPopOne',
                        fontSize: 10,
                        fontWeight: FontWeight.w400,
                        color: Colors.red,
                    ))));
    }
    return const Text(");
}
    
```

```

bool allOptionsSelected() {
    for (String question in questions) {
        if (mapSelectedOptions[question] == null) {
            return false;
        }
    }
    return true;
}

void setSelectionOptions(String question, String value) {
    mapSelectedOptions[question] = value;
    update();
}

void abortAllSelectionOptions() {
    questions.clear();
    mapSelectedOptions.clear();
    resultTest.value = 0.0;
}

final diagnoseRepositoryController = Get.put(DiagnoseRepositoryController());

Future<void> calculateDiagnose() async {
    double result = await
diagnoseRepositoryController.calculateDiagnose(mapSelectedOptions);
    int nilaiPersentase = (result * 100).toInt();
    resultTest.value = nilaiPersentase.toDouble();
    update();
}

Stream<List<Diagnose>> getAllDiagnose() {
    return diagnoseRepositoryController.getAllDiagnose();
}

```

}

diagnose_history_controller.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/diagnose_history_r
epository.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose_history.dart';
import 'package:get/get.dart';

class DiagnoseHistoryController extends GetxController {
    static DiagnoseHistoryController get instance => Get.find();

    final diagnoseHistoryRepositoryController =
Get.put(DiagnoseHistoryRepositoryController());

    Future<void> insertToDiagnoseHistory(DiagnoseHistory data) async {

        await diagnoseHistoryRepositoryController.insertToDiagnoseHistory(data);
    }

    Stream<List<Map<String, dynamic>>> getHistoryDiagnoseByUid() =>
diagnoseHistoryRepositoryController.getHistoryDiagnoseByUid();
}
```

music_admin_controller.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/music_repository.d
art';
```

```

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:flutter/material.dart';
import 'package:file_picker/file_picker.dart';
import 'package:get/get.dart';

class MusicAdminController extends GetxController {
    static MusicAdminController get instance => Get.find();

    final Rx< GlobalKey<FormState>> formKey =
        Rx< GlobalKey<FormState>>( GlobalKey<FormState>());
    Rx<PlatformFile?> imageFile = Rx<PlatformFile?>(null);
    Rx<PlatformFile?> audioFile = Rx<PlatformFile?>(null);

    Rx<TextEditingController> nameController =
        Rx<TextEditingController>(TextEditingController());
    RxMap<String, dynamic> dataMusicRequest = RxMap(
        { "music_id": "", "music_name": "", "music_url": "", "music_cover": "" });

    RxBool setEditMusic = RxBool(false);

    void clearImageAndAudio() {
        imageFile.value = null;
        audioFile.value = null;
        update();
    }

    void clearAllData() {
        nameController.value.text = "";
        audioFile.value = null;
        imageFile.value = null;
        setEditMusic.value = false;
    }
}

```

```
}
```

```

void reloadData() {
    formKey.value.currentState!.reset();
    clearAllData();
    update();
}

```

```

@Override
void onClose() {
    super.onClose();
    clearAllData();
}

```

```

Future<void> pickImage() async {
    final result = await FilePicker.platform.pickFiles(type: FileType.image);

    if (result != null) {
        imageFile.value = result.files.first;
        update();
    }
}

```

```

Future<void> pickAudio() async {
    final result = await FilePicker.platform.pickFiles(type: FileType.audio);

    if (result != null) {
        audioFile.value = result.files.first;
        update();
    }
}

```

```
final musicRepositoryController = Get.put(MusicRepositoryController());
```

```

void setIsEditVideo(Music music) {
    setEditMusic.value = true;

    nameController.value.text = music.musicName!;

    dataMusicRequest.value = {
        "music_id": music.musicId,
        "music_name": music.musicName,
        "music_url": audioFile.value,
        "music_cover": imageFile.value,
    };
}

update();
}

```

```

Future<void> editAudio() async {
    dataMusicRequest["music_name"] = nameController.value.text;
    dataMusicRequest["music_url"] = audioFile.value;
    dataMusicRequest["music_cover"] = imageFile.value;
    await musicRepositoryController
        .editMusic(dataMusicRequest)
        .whenComplete(() {
            setEditMusic.value = false;
            update();
        });
}

```

```

Future<void> uploadAudio() async {
    dataMusicRequest.value = {
        "music_id": "",
        "music_name": nameController.value.text,
        "music_url": audioFile.value,
    }
}

```

```

    "music_cover": imageFile.value,
};

await musicRepositoryController.uploadAudio(dataMusicRequest);
}

```

```

Future<void> deleteAudio(String id) async {
    await musicRepositoryController.deleteMusic(id);
}
}

```

profile_controller.dart

```

import 'dart:io';

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller
.dart';

import 'package:file_picker/file_picker.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class ProfileController extends GetxController {
    static ProfileController get instance => Get.find();

    final userGuardsController = Get.put(UserGuardsController());

    final Rx< GlobalKey<FormState>> formKeyUpdateProfile =
        Rx< GlobalKey<FormState>>( GlobalKey<FormState>());
}

final Rx< GlobalKey<FormState>> formKeyUpdateEmail =
    Rx< GlobalKey<FormState>>( GlobalKey<FormState>());

```

```

final passwordController =
    Rx<TextEditingController>(TextEditingController());

final usernameController =
    Rx<TextEditingController>(TextEditingController());

final emailController =
    Rx<TextEditingController>(TextEditingController());

Rx<PlatformFile?> imageFile = Rx<PlatformFile?>(null);

// final user = FirebaseAuth.instance.currentUser!;

Future<void> pickImage() async {
    final result = await FilePicker.platform.pickFiles(type: FileType.image);

    if (result != null) {
        imageFile.value = result.files.first;
        update();
    }
}

Future<void> updateUserProfile() async {
    final profileRef = FirebaseStorage.instance
        .ref('uploads/images/profile/${usernameController.value.text}');

    if (imageFile.value != null) {
        ListResult oldProfileRef = await FirebaseStorage.instance
            .ref('uploads/images/profile/${usernameController.value.text}')
            .listAll();

        await Future.forEach(oldProfileRef.items, (Reference ref) async {
            await ref.delete();
        });
    }
}

```

```
}).whenComplete(() => { }).catchError((e) {
    Get.snackbar("Error", "Error happen while Delete Picture");
});

final profileUrl = await profileRef.child(imageFile.value!.name).putFile(
    File(imageFile.value!.path.toString()),
    SettableMetadata(contentType: 'image/${imageFile.value!.extension}'));
}

final getProfileUrl = await profileUrl.ref.getDownloadURL();

await
userGuardsController.user.currentUser!.updateDisplayName(usernameController.value
.text);
await userGuardsController.user.currentUser!.updatePhotoURL(getProfileUrl);
}

}
```

```
qa_controller.dart

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import './repository/diagnose_repository_controller.dart';

class QaController extends GetxController {
    static QaController get instance => Get.find();
}
```

```
final RxList< GlobalKey< FormState >> formKeys =  
    RxList< GlobalKey< FormState >>([ ]);
```

```
final RxList< GlobalKey<FormState>> formSecondKeys =  
RxList< GlobalKey<FormState>>([]);
```

```
final RxList< GlobalKey<FormState>> formThirdKeys =  
RxList< GlobalKey<FormState>>([]);
```

```
final Rx<TextEditingController> testTitleController =  
Rx<TextEditingController>(TextEditingController());
```

```
final RxList<dynamic> testQa = RxList<dynamic>([]);
```

```
final RxString question = RxString("");  
final RxString keyAnswer = RxString("");  
final RxString nameAnswer = RxString("");  
final RxDouble valueAnswer = RxDouble(0.0);  
final RxDouble expertAnswer = RxDouble(0.0);
```

```
final RxBool isEdit = RxBool(false);
```

```
final RxMap<String, dynamic> testQaData = RxMap<String, dynamic>({  
"test_id": "",  
"test_title": "",  
"test_qa": [],  
});
```

```
void addTestQa() {  
testQa.add({"answer": {}, "question": ""});  
update();  
}
```

```
void removeTestQa(int index) {  
testQa.removeAt(index);
```

```
update();  
}  
  
void setQuestion(String value) {  
    question.value = value;  
    update();  
}  
  
void setKeyAnswer(String value) {  
    keyAnswer.value = value;  
    update();  
}  
  
void setNameAnswer(String value) {  
    nameAnswer.value = value;  
    update();  
}  
  
void setValueAnswer(String value) {  
    valueAnswer.value = double.parseDouble(value);  
    update();  
}  
  
void setExpertAnswer(String value) {  
    expertAnswer.value = double.parseDouble(value);  
    update();  
}  
  
void onSubmitExpertValue(int index) {  
    testQa[index]['cf_expert'] = expertAnswer.value;  
    update();  
}
```

```

void deleteAnswer(int index, String key) {
    if (testQa.length > index) {
        testQa[index]['answer'].remove(key);
        update();
    }
}

void onSubmitQuestion(int index) {
    testQa[index]['question'] = question.value;
    update();
}

void reloadData() {
    formKey.value.currentState!.reset();
    clearAllData();
    update();
}

void clearAllData() {
    testQa.value = [];
    testQaData.value = {
        "test_id": "",
        "test_title": "",
        "test_qa": null,
    };
    question.value = "";
    keyAnswer.value = "";
    nameAnswer.value = "";
    valueAnswer.value = 0.0;
    expertAnswer.value = 0.0;
    testTitleController.value.text = "";
    isEdit.value = false;
    testTitle.value = "";
}

```

```

formKeys.value = [];
formSecondKeys.value = [];
formThirdKeys.value = [];
}

@Override
void onClose() {
    super.onClose();
    clearAllData();
}

void onSubmitAnswer(int index) {
    testQa[index]['answer'][keyAnswer.value] = {
        "name": nameAnswer.value,
        "value": valueAnswer.value,
    };
    update();
}

void setEditDiagnose(Diagnose data) {
    isEdit.value = true;
    testQa.value = data.testQa;
    testTitleController.value.text = data.testTitle!;
    testQaData.value = {
        "test_id": dataTestId,
        "test_title": data.testTitle!,
        "test_qa": testQa,
    };
    update();
}

final diagnoseRepositoryController = Get.put(DiagnoseRepositoryController());
    
```

```

RxString testTitle = RxString("");


void setSearch(String value) {
    testTitle.value = value;
    update();
}

Stream<List<Diagnose>> searchDiagnose() =>
    diagnoseRepositoryController.searchDiagnose(testTitle.value);

Future<void> updateQa() async {

    testQaData["test_title"] = testTitleController.value.text;
    testQaData["test_qa"] = testQa;

    await diagnoseRepositoryController.updateQA(testQaData).then((value) {
        isEdit.value = false;
        update();
    });
}

Future<void> insertQA() async {
    testQaData.value = {
        "test_id": "",
        "test_title": testTitleController.value.text,
        "test_qa": testQa,
    };
    await diagnoseRepositoryController.insertQA(testQaData);
}

Future<void> deleteQA(String doc) async {
    await diagnoseRepositoryController.deleteQA(doc);
}

```

```
}
```

```
}
```

rule_based_controller.dart

```
import  
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/rule_based_reposit  
ory_controller.dart';  
import  
'package:aplikasi_kesehatan_mental_anak_remaja/models/rule_based_diagnose.dart';  
import 'package:flutter/material.dart';  
import 'package:get/get.dart';  
  
class RuleBasedDiagnoseController extends GetxController {  
    static RuleBasedDiagnoseController get instance => Get.find();  
  
    final Rx< GlobalKey< FormState >> formKey =  
        Rx< GlobalKey< FormState >>( GlobalKey< FormState >());  
  
    Rx< TextEditingController > ruleBasedDetailController =  
        Rx< TextEditingController >( TextEditingController());  
  
    Rx< TextEditingController > maxController =  
        Rx< TextEditingController >( TextEditingController());  
  
    Rx< TextEditingController > minController =  
        Rx< TextEditingController >( TextEditingController());  
  
    Rx< TextEditingController > nameController =  
        Rx< TextEditingController >( TextEditingController());  
  
    RxMap< String, dynamic > ruleBasedRequest = RxMap( {
```

```

    "rule_based_diagnose_id": "",  

    "rule_based_diagnose_min_percent": 0.0,  

    "rule_based_diagnose_max_percent": 0.0,  

    "rule_based_diagnose_name": "",  

});  
  

void clearAllData() {  

    nameController.value.text = "";  

    minController.value.text = "";  

    maxController.value.text = "";  

    ruleBasedDetailController.value.text = "";  

    setEdit.value = false;  

    ruleBasedName.value = "";  

}  
  

void reloadData() {  

    formKey.value.currentState!.reset();  

    clearAllData();  

    update();  

}  
  

@Override  

void onClose() {  

    super.onClose();  

    clearAllData();  

}  
  

RxBool setEdit = RxBool(false);  
  

RuleBaseDiagnoseRepositoryController ruleBaseDiagnoseRepositoryController =  

    Get.put(RuleBaseDiagnoseRepositoryController());  
  

void setEditRuleBased(RuleBasedDiagnose rulebased) {

```

```

setEdit.value = true;

maxController.value.text = rulebased.ruleBasedDiagnoseMaxPercent.toString();
minController.value.text = rulebased.ruleBasedDiagnoseMinPercent.toString();
nameController.value.text = rulebased.ruleBasedDiagnoseName!;
ruleBasedDetailController.value.text = rulebased.ruleBasedDiagnoseDetail!;

ruleBasedRequest.value = {
    "rule_based_diagnose_id": rulebased.ruleBasedDiagnoseId,
    "rule_based_diagnose_min_percent":
        rulebased.ruleBasedDiagnoseMinPercent,
    "rule_based_diagnose_max_percent":
        rulebased.ruleBasedDiagnoseMaxPercent,
    "rule_based_diagnose_name": rulebased.ruleBasedDiagnoseName,
    "rule_based_diagnose_detail": rulebased.ruleBasedDiagnoseDetail
};

update();
}

Future<void> updateRuleBased() async {
    ruleBasedRequest["rule_based_diagnose_min_percent"] =
        minController.value.text;
    ruleBasedRequest["rule_based_diagnose_max_percent"] =
        maxController.value.text;
    ruleBasedRequest["rule_based_diagnose_name"] = nameController.value.text;
    ruleBasedRequest["rule_based_diagnose_detail"] =
        ruleBasedDetailController.value.text;

    await ruleBaseDiagnoseRepositoryController
        .updateRuleBased(ruleBasedRequest)
        .then((value) {
            setEdit.value = false;
        });
}

```

```

        update();
    });
}

Future<void> deleteRuleBased(String id) async {
    await ruleBaseDiagnoseRepositoryController.deleteRuleBased(id);
}

Future<void> insertRuleBased() async {
    ruleBasedRequest.value = {
        "rule_based_diagnose_id": "",
        "rule_based_diagnose_min_percent": double.parse(minController.value.text),
        "rule_based_diagnose_max_percent": double.parse(maxController.value.text),
        "rule_based_diagnose_detail": ruleBasedDetailController.value.text,
        "rule_based_diagnose_name": nameController.value.text,
    };
    await ruleBaseDiagnoseRepositoryController
        .insertRuleBased(ruleBasedRequest);
}

RxString ruleBasedName = RxString("");

void setSearch(String value) {
    ruleBasedName.value = value;
    update();
}

Stream<List<RuleBasedDiagnose>> searchRuleBased() =>
    ruleBaseDiagnoseRepositoryController.searchRuleBased(ruleBasedName.value);

Stream<List<RuleBasedDiagnose>> getAllRuleBased() =>
    ruleBaseDiagnoseRepositoryController.getAllRuleBased();

```

```

Stream<Map<String, dynamic>> getRuleBasedDiagnose() =>
    ruleBaseDiagnoseRepositoryController.getRuleBasedDiagnose();
}

```

video_admin_controller.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/video_repository_c
ontroller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import 'package:flutter/material.dart';
import 'package:file_picker/file_picker.dart';
import 'package:get/get.dart';

class VideoAdminController extends GetxController {
    static VideoAdminController get instance => Get.find();

    final Rx< GlobalKey<FormState>> formKey =
        Rx< GlobalKey<FormState>>( GlobalKey<FormState>());
    Rx<PlatformFile?> imageFile = Rx<PlatformFile?>(null);
    Rx<PlatformFile?> videoFile = Rx<PlatformFile?>(null);

    Rx<TextEditingController> titleController =
        Rx<TextEditingController>(TextEditingController());
    Rx<TextEditingController> descriptionController =
        Rx<TextEditingController>(TextEditingController());

    RxMap<String, dynamic> dataVideoRequest = RxMap({
        "video_id": "",
        "video_title": "",
        "video_description": ""
    });
}

```

```
"video_url": "",  
"video_caption_url": ""  
});  
  
RxBool setEditVideo = RxBool(false);  
  
void clearImageAndVideo() {  
    imageFile.value = null;  
    videoFile.value = null;  
    update();  
}  
  
@override  
void onClose() {  
    super.onClose();  
    clearAllData();  
}  
  
void clearAllData() {  
    titleController.value.text = "";  
    descriptionController.value.text = "";  
    videoFile.value = null;  
    imageFile.value = null;  
    setEditVideo.value = false;  
}  
  
void reloadData() {  
    formKey.value.currentState!.reset();  
    clearAllData();  
    update();  
}
```

```

Future<void> pickImage() async {
    final result = await FilePicker.platform.pickFiles(type: FileType.image);

    if (result != null) {
        imageFile.value = result.files.first;
    }
    update();
}

Future<void> pickVideo() async {
    final result = await FilePicker.platform.pickFiles(type: FileType.video);

    if (result != null) {
        videoFile.value = result.files.first;
    }
    update();
}

final videoRepositoryController = Get.put(VideoRepositoryController());

void setIsEditVideo(Video video) {
    setEditVideo.value = true;

    titleController.value.text = video.videoTitle!;
    descriptionController.value.text = video.videoDescription!;

    dataVideoRequest.value = {
        "video_id": video.videoId,
        "video_title": video.videoTitle,
        "video_description": video.videoDescription,
        "video_url": videoFile.value,
        "video_caption_url": imageFile.value
    };
}

```

```
        update();
```

```
}
```

```
Future<void> editVideo() async {
    dataVideoRequest["video_title"] = titleController.value.text;
    dataVideoRequest["video_description"] = descriptionController.value.text;
    dataVideoRequest["video_url"] = videoFile.value;
    dataVideoRequest["video_caption_url"] = imageFile.value;
    await videoRepositoryController
        .editVideo(dataVideoRequest)
        .whenComplete(() {
            setEditVideo.value = false;
            update();
        });
}
```

```
Future<void> uploadVideo() async {
    dataVideoRequest.value = {
        "video_id": "",
        "video_title": titleController.value.text,
        "video_description": descriptionController.value.text,
        "video_url": videoFile.value,
        "video_caption_url": imageFile.value
    };
    await videoRepositoryController.uploadVideo(dataVideoRequest);
}
```

```
Future<void> deleteVideo(String id) async {
    await videoRepositoryController.deleteVideo(id);
}
```

video_controller.dart

```
import 'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/video_repository_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/video_watch_screen.dart';
import 'package:flutter/services.dart';
import 'package:get/get.dart';
import 'package:video_player/video_player.dart';
import 'package:wakelock/wakelock.dart';

class VideoController extends GetxController {
    static VideoController get instance => Get.find();

    RxString urlVideo = RxString("");
    RxBool isVideoPlaying = RxBool(false);
    Rx<Duration> duration = Rx<Duration>(Duration.zero);
    Rx<Duration> position = Rx<Duration>(Duration.zero);
    RxBool setFullScreen = RxBool(false);
    RxBool setLandscapeOrPortrait = RxBool(false);
    RxBool onShowPosition = RxBool(false);

    late VideoPlayerController videoPlayerController;
    late Future<void> initializeVideoPlayerFuture;

    void setFullScreenVideo() {
        setFullScreen.value = !setFullScreen.value;
        update();
    }
}
```

```

void setVideo(Video video) {
    videoPlayerController =
        VideoPlayerController.networkUrl(Uri.parse(video.videoUrl));
    initializeVideoPlayerFuture = videoPlayerController.initialize();
    videoPlayerController.addListener(() {
        if (videoPlayerController.value.hasError) {
            Get.back();
        }
        if (videoPlayerController.value.isInitialized) {
            duration.value = videoPlayerController.value.duration;
            position.value = videoPlayerController.value.position;
            isVideoPlaying.value = videoPlayerController.value.isPlaying;
            Get.to(VideoWatchScreen(video));
        }
    });
    update();
}

void onClickEvent() {
    videoPlayerController.value.isPlaying
        ? videoPlayerController.pause()
        : videoPlayerController.play();
    update();
}

void slider(double value) {
    videoPlayerController.seekTo(Duration(seconds: value.toInt()));
    videoPlayerController.play();
    update();
}

String formatTimeVideo(int seconds) {
    return '${(Duration(seconds: seconds))}'.split('.')[0].padLeft(8, '0');
}

```

```
}
```

```

void disposeVideoPlayer() async {
    videoPlayerController.dispose();
    setFullScreen.value = false;
    onShowPosition.value = false;
    setLandscapeOrPortrait.value = false;
    videoTitle.value = "";
    await SystemChrome.setPreferredOrientations(DeviceOrientation.values);
    await Wakelock.disable();
}
```

```
final videoRepositoryController = Get.put(VideoRepositoryController());
```

```
RxString videoTitle = RxString("");
```

```

void clearSearch() {
    videoTitle.value = "";
    update();
}
```

```

void setSearch(String value) {
    videoTitle.value = value;
    update();
}
```

```

Stream<List<Video>> searchVideo() =>
videoRepositoryController.searchVideo(videoTitle.value);
```

```

Stream<List<Video>> getAllVideo() => videoRepositoryController.getAllVideo();
}
```

Bagian Repository

```
authentication_repository_controller.dart

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller
.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/utils/exceptions/signin_exception.da
rt';
import
'package:aplikasi_kesehatan_mental_anak_remaja/utils/exceptions/signup_exception.da
rt';
import 'package:get/get.dart';
import 'package:firebase_auth/firebase_auth.dart';

class AuthenticationRepositoryController extends GetxController {
    static AuthenticationRepositoryController get instance => Get.find();

    final _userGuardsController = Get.put(UserGuardsController());

    final _auth = FirebaseAuth.instance;

    Future<void> signInWithEmailAndPassword(String email, String password) async {
        try {
            await _auth.signInWithEmailAndPassword(email: email, password: password);
        } on FirebaseAuthException catch (e) {
            final err = SignInWithEmailAndPasswordFailureException.code(e.code);
            throw err;
        } catch (e) {
            final err = SignUpWithEmailAndPasswordFailureException();
            throw err;
        }
    }
}
```

```
}
```

```

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

Future<void> createUserWithEmailAndPassword(
  String email, String password, String username) async {
  try {
    final user = await _auth.createUserWithEmailAndPassword(
      email: email, password: password);
    if (user.additionalUserInfo!.isNewUser) {
      await _userGuardsController.user.currentUser
        ?.updateDisplayName(username);
    }
  } on FirebaseAuthException catch (e) {
    final err = SignUpWithEmailAndPasswordFailureException.code(e.code);
    throw err;
  } catch (e) {
    final err = SignUpWithEmailAndPasswordFailureException();
    throw err;
  }
}

```

crisis_support_controller_repository.dart

```

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/crisis_support.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

```

```

import 'package:get/get.dart';

class CrisisSupportRepositoryController extends GetxController {
    static CrisisSupportRepositoryController get instance => Get.find();

    final crisisSupportRepo =
        FirebaseFirestore.instance.collection('CrisisSupport');

    Stream<List<CrisisSupport>> getAllCrisisSupport() => crisisSupportRepo
        .snapshots()
        .map((e) => e.docs.map((e) => CrisisSupport.fromSnapshot(e)).toList());
}

```

diagnose_history_repository.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_controller
.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller
.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/repository/rule_based_reposit
ory_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose_history.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/landing_screen.dar
t';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:get/get.dart';

class DiagnoseHistoryRepositoryController extends GetxController {

```

```

static DiagnoseHistoryRepositoryController getInstance => Get.find();

final diagnoseRepo = FirebaseFirestore.instance.collection('DiagnoseHistory');
final diagnoseRepoWithId =
    FirebaseFirestore.instance.collection('DiagnoseHistory').doc();
final ruleBasedDiagnose = Get.put(RuleBaseDiagnoseRepositoryController());
final diagnoseController = Get.put(DiagnoseController());
final userGuardsController = Get.put(UserGuardsController());

```

```

Future<void> insertToDiagnoseHistory(DiagnoseHistory data) async {
    data.diagnoseHistoryId = diagnoseRepoWithId.id;
    await diagnoseRepoWithId.set(data.toMap()).whenComplete(() {
        Get.snackbar("Completed", "Success Save Diagnose");
        diagnoseController.abortAllSelectionOptions();
        Get.offAll(LandingScreen());
    }).catchError((error, stackTrace) {
        Get.snackbar("Error", "Error Happen While Saving Data");
    });
}

```

```

Stream<List<Map<String, dynamic>>> getHistoryDiagnoseByUid() {
    final uId = userGuardsController.user.currentUser?.uid;

    return diagnoseRepo
        .where("user_id", isEqualTo: uId ?? "")
        .snapshots().asyncMap((event) async {
            List<Map<String, dynamic>> result = [];
            for(var e in event.docs) {
                Map<String, dynamic> mapToDiagnoseHistory = {
                    "diagnose_history_date": e.data()["diagnose_history_date"],
                    "diagnose_history_id": e.data()["diagnose_history_id"],
                    "diagnose_history_user_answer": e.data()["diagnose_history_user_answer"],
                };
            }
        });
}

```

```

    "diagnose_history_level_depression":  

e.data()["diagnose_history_level_depression"],  

    };  

    var snapshot = await  

ruleBasedDiagnose.ruleBasedDiagnoseRepo.where("rule_based_diagnose_id",  

isEqualTo: e.data()["diagnose_history_rulebased_id"]).get();  

    for (var element in snapshot.docs) {  

        mapToDiagnoseHistory["diagnose_history_name"] =  

element.data()["rule_based_diagnose_name"];  

        mapToDiagnoseHistory["diagnose_history_detail"] =  

element.data()["rule_based_diagnose_detail"];  

    }  

    result.add(mapToDiagnoseHistory);  

}  

return result;  

});  

}
}

```

diagnose_repository_controller.dart

```

import 'dart:convert';  

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';  

import 'package:cloud_firestore/cloud_firestore.dart';  

import 'package:get/get.dart';  

class DiagnoseRepositoryController extends GetxController {  

    static DiagnoseRepositoryController get instance => Get.find();  

    final diagnoseRepo = FirebaseFirestore.instance.collection('Diagnose');

```

```

Future<double> getInitialCFExpert() async {
    double cfTotal = 0.0;

    await diagnoseRepo.snapshots().first.then((QuerySnapshot event) async {
        if (event.docs.isNotEmpty) {
            QueryDocumentSnapshot firstDocument = event.docs.first;
            cfTotal = firstDocument.get('test_qa')[0]?"cf_expert"] ?? 0.0;
        }
    });
}

return cfTotal;
}

double combineCF(double currentCF, double nextCF) {
    return currentCF + nextCF * (1 - currentCF);
}

Future<double> calculateDiagnose(Map<String, dynamic> data) async {
    List<double> cfExpert = [];
    List<double> cfUser = [];
    List<double> cfCombination = [];

    // double cfTotal = 0.0;

    var snapshot = await diagnoseRepo.snapshots().first;
    for (var expertDiagnoseData in snapshot.docs) {
        cfExpert
            .add(expertDiagnoseData.data()["test_qa"][0]["cf_expert"].toDouble());
    }

    for (var diagnoseData in data.values) {
        cfUser.add(jsonDecode(diagnoseData)["value"].toDouble());
    }
}

```

```

for (int index = 0; index < cfExpert.length; index++) {
    cfCombination.add(cfExpert[index] * cfUser[index]);
}

double cfTotal = await getInitialCFExpert() *
    jsonDecode(data.entries.first.value)["value"].toDouble() +
    (cfCombination[1] *
        (1 -
            (await getInitialCFExpert() *
                jsonDecode(data.entries.first.value)["value"].toDouble())));

int next = 2;
for (int index = 0; index < cfCombination.length; index++) {
    if (next > cfCombination.length - 1) {
        break;
    }
    double currentCF = cfTotal;
    double nextCF = cfCombination[next];
    cfTotal = currentCF + nextCF * (1 - currentCF);
    cfCombination[next] = cfTotal;
    next++;
    /**
     print('Iterasi ke-$index');
     print('Nilai awal: $currentCF');
     print('Nilai berikutnya: $nextCF');
     print('CF Total setelah penggabungan: $cfTotal');

     print(
         'proses perhitungan: ${cfTotal += cfCombination[index] + (cfCombination[next]
         * (1 - cfCombination[index]))}');
     cfTotal += cfCombination[index] +
         (cfCombination[next] * (1 - cfCombination[index])));
}

```

```
**/
```

```
}
```

```
return cfTotal;
```

```
}
```

```
Stream<List<Diagnose>> searchDiagnose(String testTitle) {
    if (testTitle.isEmpty) {
        return diagnoseRepo.snapshots().map((snapshot) =>
            snapshot.docs.map((doc) => Diagnose.fromSnapshot(doc)).toList());
    } else {
        return diagnoseRepo.snapshots().map((snapshot) => snapshot.docs
            .where((doc) {
                String sqlLikePattern = ".${$RegExp.escape(testTitle)}.*";
                RegExp pattern = RegExp(sqlLikePattern, caseSensitive: false);
                return pattern.hasMatch(doc["test_title"]);
            })
            .map((doc) => Diagnose.fromSnapshot(doc))
            .toList());
    }
}
```

```
Future<void> deleteQA(String doc) async {
    await diagnoseRepo.doc(doc).delete().whenComplete(() {
        Get.snackbar("Completed", "Success Delete Diagnose");
    }).catchError((error, stackTrace) {
        Get.snackbar("Error", "Error While Delete Diagnose");
    });
}
```

```
Future<void> updateQA(Map<String, dynamic> data) async {
    await diagnoseRepo.doc(data['test_id']).update(data).whenComplete(() {
```

```

        Get.snackbar("Completed", "Success Update Diagnose");
    }).catchError((error, stackTrace) {
        Get.snackbar("Error", "Error While Update Diagnose");
    });
}

Future<void> insertQA(Map<String, dynamic> data) async {
    final newId = FirebaseFirestore.instance.collection('Diagnose').doc();
    data["test_id"] = newId.id;
    await newId.set(data).whenComplete(() {
        Get.snackbar("Completed", "Success Insert Diagnose");
    }).catchError((error, stackTrace) {
        Get.snackbar("Error", "Error While Insert Diagnose");
    });
}

Stream<List<Diagnose>> getAllDiagnose() => diagnoseRepo.snapshots().map((e) {
    return e.docs.map((e) => Diagnose.fromSnapshot(e)).toList();
});
}

```

music_repository.dart

```

import 'dart:io';

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:get/get.dart';

class MusicRepositoryController extends GetxController {
    static MusicRepositoryController get instance => Get.find();

    final musicRepo = FirebaseFirestore.instance.collection('Music');

```

```

Stream<List<Music>> searchMusic(String musicName) {
    if (musicName.isEmpty) {
        return musicRepo
            .snapshots()
            .map((e) => e.docs.map((e) => Music.fromSnapshot(e)).toList());
    } else {
        return musicRepo
            .snapshots()
            .map((snapshot) => snapshot.docs
                .where((doc) {
                    String sqlLikePattern = ".*${RegExp.escape(musicName)}.*";
                    RegExp pattern = RegExp(sqlLikePattern, caseSensitive: false);
                    return pattern.hasMatch(doc["music_name"]);
                })
                .map((doc) => Music.fromSnapshot(doc))
                .toList());
    }
}

```

```

Future<void> editMusic(Map<String, dynamic> data) async {
    try {
        final audioRef =
            FirebaseStorage.instance.ref('uploads/musics/${data["music_id"]}');

        final imageRef =
            FirebaseStorage.instance.ref('uploads/images/${data["music_id"]}');
        if (data['music_cover'] != "" ||
            data['music_cover'] != null && data['music_url'] != "" ||
            data['music_url'] != null) {
            ListResult oldMusicRef = await FirebaseStorage.instance
                .ref('uploads/musics/${data['music_id']}')

```

```

.listAll();

ListResult oldImageRef = await FirebaseStorage.instance
    .ref('uploads/images/${data['music_id']}')
.listAll();

await Future.forEach(oldMusicRef.items, (Reference ref) async {
    await ref.delete();
}).whenComplete(() => {}).catchError((e) {
    Get.snackbar(
        "Error", "Error While Delete Music");
});

await Future.forEach(oldImageRef.items, (Reference ref) async {
    await ref.delete();
}).whenComplete(() => {}).catchError((e) {
    Get.snackbar(
        "Error", "Error While Delete Image");
});

final newMusicUrl = await audioRef
    .child(data['music_url'].name)
.putFile(
    File(data['music_url'].path),
    SettableMetadata(
        contentType: 'audio/${data['music_url'].extension}');
);

final newImageUrl = await imageRef
    .child(data['music_cover'].name)
.putFile(
    File(data['music_cover'].path),
    SettableMetadata(
        contentType: 'image/${data['music_cover'].extension}');
);

data['music_url'] = await newMusicUrl.ref.getDownloadURL();
data['music_cover'] = await newImageUrl.ref.getDownloadURL();

```

```
}
```

```

await musicRepo.doc(data['music_id']).update(data).whenComplete(() {
    Get.snackbar("Completed", "Update Music Success");
}).catchError((e) {
    Get.snackbar(
        "Error", "Error While Updating Music");
});
} catch (e) {
    Get.snackbar("Error", "Error While Updating Music");
}
}
}
```

```

Future<void> deleteMusic(String id) async {
try {
ListResult audioRef =
    await FirebaseStorage.instance.ref('uploads/musics/$id').listAll();
ListResult imageRef =
    await FirebaseStorage.instance.ref('uploads/images/$id').listAll();

await Future.forEach(audioRef.items, (Reference ref) async {
    await ref.delete();
}).whenComplete(() => {}).catchError((e) {
    Get.snackbar("Error", "Error While Delete Music");
});

await Future.forEach(imageRef.items, (Reference ref) async {
    await ref.delete();
}).whenComplete(() => {}).catchError((e) {
    Get.snackbar("Error", "Error While Delete Image");
});

await musicRepo.doc(id).delete().whenComplete(() {
```

```

Get.snackbar("Completed", "Succes Delete Music");
}).catchError((e) {
    Get.snackbar("Error", "Error While Delete Music");
});
} catch (e) {
    Get.snackbar("Error", "Error While Delete Music");
}
}

Future<void> uploadAudio(Map<String, dynamic> data) async {
try {
    final musicRepoWithId = FirebaseFirestore.instance.collection('Music').doc();
    data['music_id'] = musicRepoWithId.id;
    final musicRef =
        FirebaseStorage.instance.ref('uploads/musics/${musicRepoWithId.id}/');
    final imageRef =
        FirebaseStorage.instance.ref('uploads/images/${musicRepoWithId.id}/');

    final musicUrl = await musicRef.child(data['music_url'].name).putFile(
        File(data['music_url'].path),
        SettableMetadata(
            contentType: 'audio/${data['music_url'].extension}');
    );

    final imageUrl = await imageRef.child(data['music_cover'].name).putFile(
        File(data['music_cover'].path),
        SettableMetadata(
            contentType: 'image/${data['music_cover'].extension}');
    );

    data['music_url'] = await musicUrl.ref.getDownloadURL();
    data['music_cover'] = await imageUrl.ref.getDownloadURL();

    await musicRepoWithId.set(data).whenComplete(() {
        Get.snackbar("Success", "Success Add Music");
    });
}
}

```

```

}).catchError((e) {
    Get.snackbar("Error", "Error While Add Music ");
});
} on FirebaseException {
    Get.snackbar("Error", "Error While Add Music");
}
}

Stream<List<Music>> getAllMusic() => musicRepo
    .snapshots()
    .map((e) => e.docs.map((e) => Music.fromSnapshot(e)).toList());
}

```

rule_based_repository_controller.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/models/rule_based_diagnose.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:get/get.dart';

class RuleBaseDiagnoseRepositoryController extends GetxController {
    static RuleBaseDiagnoseRepositoryController get instance => Get.find();

    final ruleBasedDiagnoseRepo =
        FirebaseFirestore.instance.collection('RuleBasedDiagnose');
    final diagnoseController = Get.put(DiagnoseController());

    Stream<List<RuleBasedDiagnose>> searchRuleBased(String ruleBasedName) {
        if (ruleBasedName.isEmpty) {
            return ruleBasedDiagnoseRepo.snapshots().map((snapshot) => snapshot.docs

```

```

.map((doc) => RuleBasedDiagnose.fromSnapshot(doc))
.toList());
} else {
    return ruleBasedDiagnoseRepo.snapshots().map((snapshot) => snapshot.docs
        .where((doc) {
            String sqlLikePattern = ".*${RegExp.escape(ruleBasedName)}.*";
            RegExp pattern = RegExp(sqlLikePattern, caseSensitive: false);
            return pattern.hasMatch(doc["rule_based_diagnose_name"]);
        })
        .map((doc) => RuleBasedDiagnose.fromSnapshot(doc))
        .toList());
}
}

```

```

Future<void> deleteRuleBased(String id) async {
    await ruleBasedDiagnoseRepo.doc(id).delete().whenComplete(() {
        Get.snackbar("Completed", "Success Delete Rule Based");
    }).catchError((e) {
        Get.snackbar("Error", "Error Delete Rule Based");
    });
}

```

```

Future<void> updateRuleBased(Map<String, dynamic> data) async {
    await ruleBasedDiagnoseRepo
        .doc(data["rule_based_diagnose_id"])
        .update(data)
        .whenComplete(() {
            Get.snackbar("Success", "Success Update Rule Based");
        }).catchError((e) {
            Get.snackbar("Error", "Error While Update Rule Based");
        });
}

```

```

Future<void> insertRuleBased(Map<String, dynamic> data) async {
    final ruleBasedDiagnoseRepoWithId =
        FirebaseFirestore.instance.collection('RuleBasedDiagnose').doc();
    data['rule_based_diagnose_id'] = ruleBasedDiagnoseRepoWithId.id;

    await ruleBasedDiagnoseRepoWithId.set(data).whenComplete(() {
        Get.snackbar("Success", "Success Add Rule Based");
    }).catchError((e) {
        Get.snackbar("Error", "Error While Add Rule Based");
    });
}

Stream<List<RuleBasedDiagnose>> getAllRuleBased() =>
    ruleBasedDiagnoseRepo.snapshots().map(
        (e) => e.docs.map((e) => RuleBasedDiagnose.fromSnapshot(e)).toList());

Stream<Map<String, dynamic>> getRuleBasedDiagnose() {
    return ruleBasedDiagnoseRepo.snapshots().map((event) {
        return event.docs.fold<Map<String, dynamic>>({ },
            (Map<String, dynamic> filteredData, doc) {
                if (diagnoseController.resultTest.value >=
                    doc['rule_based_diagnose_min_percent'] &&
                    diagnoseController.resultTest.value <=
                    doc['rule_based_diagnose_max_percent']) {
                    RuleBasedDiagnose diagnose = RuleBasedDiagnose.fromQuerySnapshot(doc);
                    filteredData.addAll(diagnose.toMap());
                }
            }
        );
    });
}
}

```

video_repository_controller.dart

```

import 'dart:io';

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:get/get.dart';

class VideoRepositoryController extends GetxController {
    static VideoRepositoryController get instance => Get.find();

    final videoRepo = FirebaseFirestore.instance.collection('Video');

    Stream<List<Video>> searchVideo(String videoTitle) {
        if (videoTitle.isEmpty) {
            return videoRepo.snapshots().map((snapshot) =>
                snapshot.docs.map((doc) => Video.fromSnapshot(doc)).toList());
        } else {
            return videoRepo.snapshots().map((snapshot) => snapshot.docs
                .where((doc) {
                    String sqlLikePattern = ".${$RegExp.escape(videoTitle)}.*";
                    RegExp pattern = RegExp(sqlLikePattern, caseSensitive: false);
                    return pattern.hasMatch(doc["video_title"]);
                })
                .map((doc) => Video.fromSnapshot(doc))
                .toList());
        }
    }

    Future<void> editVideo(Map<String, dynamic> data) async {
        try {
            final videoRef =

```

```

FirebaseStorage.instance.ref('uploads/videos/${data["video_id"]}');
final imageRef =
    FirebaseStorage.instance.ref('uploads/images/${data["video_id"]}'');

if (data['video_url'] != "" ||
    data['video_url'] != null && data['video_caption_url'] != "" ||
    data['video_caption_url'] != null) {
    ListResult oldVideoRef = await FirebaseStorage.instance
        .ref('uploads/videos/${data['video_id']}')
        .listAll();

    ListResult oldImageRef = await FirebaseStorage.instance
        .ref('uploads/images/${data['video_id']}')
        .listAll();

    await Future.forEach(oldVideoRef.items, (Reference ref) async {
        await ref.delete();
    }).whenComplete(() => {}).catchError((e) {
        Get.snackbar("Error", "Error While Update Video");
    });

    await Future.forEach(oldImageRef.items, (Reference ref) async {
        await ref.delete();
    }).whenComplete(() => {}).catchError((e) {
        Get.snackbar("Error", "Error While Delete Picture");
    });
}

final newVideoUrl = await videoRef
    .child(data['video_url'].name)
    .putFile(File(data['video_url'].path),
        SettableMetadata(contentType: 'video/mp4'));

final newImageUrl = await imageRef
    .child(data['video_caption_url'].name)
    .putFile(
        File(data['video_caption_url'].path),
        SettableMetadata(

```

```

contentType:  

'image/${data['video_caption_url'].extension}');
```

```

data['video_url'] = await newVideoUrl.ref.getDownloadURL();  

data['video_caption_url'] = await newImageUrl.ref.getDownloadURL();  

}
```

```

await videoRepo.doc(data['video_id']).update(data).whenComplete(() {  

    Get.snackbar("Success", "Success Update Video");  

}).catchError((e) {  

    Get.snackbar("Error", "Error While Update Video");  

});  

} catch (e) {  

    Get.snackbar("Error", "Error While Update Video");  

}  

}
}

Future<void> deleteVideo(String id) async {
    final videoRef = FirebaseStorage.instance.ref('uploads/videos/$id');
    final imageRef = FirebaseStorage.instance.ref('uploads/images/$id');
    try {
        ListResult oldVideoRef = await videoRef.listAll();
        ListResult oldImageRef = await imageRef.listAll();

        await Future.forEach(oldVideoRef.items, (Reference ref) async {
            await ref.delete();
        }).whenComplete(() => {}).catchError((e) {
            Get.snackbar("Error", "Error While Delete Video");
        });
        await Future.forEach(oldImageRef.items, (Reference ref) async {
            await ref.delete();
        }).whenComplete(() => {}).catchError((e) {
            Get.snackbar("Error", "Error While Update Picture");
        });
    }
}

```

});

```
await videoRepo.doc(id).delete().whenComplete(() =>
    Get.snackbar("Success", "Success Delete Video");
}).catchError((e) {
    Get.snackbar("Error", "Error While Delete Video");
});
} catch (e) {
    Get.snackbar("Error", "Error While Delete Video");
}
}
```

```
Future<void> uploadVideo(Map<String, dynamic> data) async {
    try {
        final videoRepoWithId =
            FirebaseFirestore.instance.collection('Video').doc();
        data['video_id'] = videoRepoWithId.id;

        final videoRef =
            FirebaseStorage.instance.ref('uploads/videos/${videoRepoWithId.id}/');
        final imageRef =
            FirebaseStorage.instance.ref('uploads/images/${videoRepoWithId.id}/');

        final videoUrl = await videoRef.child(data['video_url'].name).putFile(
            File(data['video_url'].path),
            SettableMetadata(contentType: 'video/mp4'));
        final imageUrl = await imageRef
            .child(data['video_caption_url'].name)
            .putFile(
                File(data['video_caption_url'].path),
                SettableMetadata(
                    contentType: 'image/${data['video_caption_url'].extension}'));
    }
}
```

```

data['video_url'] = await videoUrl.ref.getDownloadURL();
data['video_caption_url'] = await imageUrl.ref.getDownloadURL();

await videoRepoWithId.set(data).whenComplete(() {
    Get.snackbar("Success", "Success Add Video");
}). catchError((e) {
    Get.snackbar("Error", "Error While Add Video");
});
} on FirebaseException {
    Get.snackbar("Error", "Error While Add Video");
}
}

Stream<List<Video>> getAllVideo() => videoRepo
    .snapshots()
    .map((e) => e.docs.map((e) => Video.fromSnapshot(e)).toList());
}

```

Bagian tampilan

A) Admin

Landing Screen

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/auth_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/admin_screen/music_screen.dart';

```

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/view/admin_screen/qa_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/admin_screen/rule_based_adm
in_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/admin_screen/video_screen.da
rt';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class AdminLandingScreen extends StatelessWidget {
AdminLandingScreen({super.key});

final List<Map<String, dynamic>> items = [
{'name': 'Music', 'route': MusicAdminScreen()},
{'name': 'Video', 'route': VideoAdminScreen()},
{'name': 'Question Answer', 'route': QuestionAnswerAdminScreen()},
{'name': 'Rule Based', 'route': RuleBasedDiagnoseAdminScreen()},
];

final authController = Get.put(AuthController());

final userGuardController = Get.put(UserGuardsController());

@Override
Widget build(BuildContext context) {
return Scaffold(
appBar: AppBar(
title: Text('Hi ${userGuardController.user.currentUser!.displayName}'),
actions: [
IconButton(
onPressed: () async => authController.logout(),

```

```
        icon: const Icon(Icons.logout))  
    ],  
    ),  
    body: GridView.builder(  
        itemCount: items.length,  
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(  
            crossAxisCount: 2,  
            crossAxisSpacing: 10.0,  
            mainAxisSpacing: 10.0,  
        ),  
        itemBuilder: (BuildContext context, int index) {  
            return InkWell(  
                onTap: () {  
                    Get.to(items[index]['route']);  
                },  
                child: Card(  
                    elevation: 5,  
                    child: Center(  
                        child: Text(  
                            items[index]['name'],  
                            style: const TextStyle(  
                                fontSize: 20.0,  
                                fontWeight: FontWeight.bold,  
                            ),  
                        ),  
                    ),  
                ),  
            );  
        },  
    );  
},  
);  
}  
}
```

```
manage_music_admin_screen.dart

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/audio_controller.d
art';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

import '../../get_x/controllers/music_admin_controller.dart';

class MusicAdminScreen extends StatelessWidget {
    final musicAdminController = Get.put(MusicAdminController());
    final musicController = Get.put(MusicController());

    MusicAdminScreen({super.key});

    @override
    Widget build(BuildContext context) {
        return GetBuilder<MusicAdminController>(
            init: musicAdminController,
            builder: (controller) {
                return WillPopScope(
                    onWillPop: () async {
                        controller.clearAllData();
                        musicController.clearSearch();
                        return true;
                    },
                    child: Scaffold(
                        appBar: AppBar(
                            leading: Builder(builder: (BuildContext context) {
```

```

return IconButton(
    onPressed: () {
        controller.onClose();
        Get.back();
    },
    icon: const Icon(Icons.arrow_back));
),

title: const Text('Music'),
),

body: Padding(
    padding: const EdgeInsets.all(20.0),
    child: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
            Form(
                key: controller.formKey.value,
                child: Column(children: [
                    TextFormField(
                        validator: (value) {
                            if (value == null || value.isEmpty) {
                                return 'Please enter some text';
                            }
                            return null;
                        },
                        controller: controller.nameController.value,
                        decoration: const InputDecoration(
                            labelText: 'Music Name'),
                ),
                const SizedBox(height: 20),
                SizedBox(
                    width: double.infinity,
                    child: ElevatedButton(
                        onPressed: controller.pickImage,

```

```

        child: const Text('Pick Cover'),
      )),  

      const SizedBox(height: 10),  

      controller.imageFile.value != null  

        ? Text(controller.imageFile.value!.name)  

        : Container(),  

      const SizedBox(height: 20),  

      SizedBox(  

        width: double.infinity,  

        child: ElevatedButton(  

          onPressed: controller.pickAudio,  

          child: const Text('Pick Music'),  

        )),  

      const SizedBox(height: 10),  

      controller.audioFile.value != null  

        ? Text(controller.audioFile.value!.name)  

        : Container(),  

      const SizedBox(height: 20),  

      Row(  

        mainAxisAlignment:  

          MainAxisAlignment.spaceBetween,  

        children: [  

          ElevatedButton(  

            onPressed: () async {  

              if (controller  

                .formKey.value.currentState!  

                .validate() {  

                if (controller.imageFile.value !=  

                  null ||  

                  controller.audioFile.value !=  

                  null) {  

                  if (controller  

                    .setEditMusic.value) {  


```

```

        await controller
            .editAudio()
            .then((value) {
                controller.resetData();
            });
        } else {
            await controller
                .uploadAudio()
                .then((value) {
                    controller.resetData();
                });
        }
    } else {
        Get.snackbar("Empty Field",
            "Image And Music must be fill");
    }
}
},
),
child: Text(
    controller.setEditMusic.value
    ? 'Update'
    : 'Add'),
),
ElevatedButton(
    onPressed: () => controller.resetData(),
    child: const Text('Reset Data'),
),
],
),
]),
Padding(
    padding: const EdgeInsets.all(8.0),
    child: TextField(

```

```

onChanged: (value) {
    musicController.setSearch(value);
},
decoration: const InputDecoration(
    labelText: 'Search',
    hintText: 'Enter search term',
    prefixIcon: Icon(Icons.search),
),
),
),
Expanded(
child: Obx(
() {
    return StreamBuilder<List<Music>>(
        stream: musicController.searchMusic(),
        builder: (context, snapshot) {
            if (snapshot.connectionState ==
                ConnectionState.waiting) {
                return const Center(
                    child: CircularProgressIndicator(),
                );
            }
            if (snapshot.hasError) {
                return Center(
                    child: Text(
                        'Error: ${snapshot.error}'),
                );
            }
            if (!snapshot.hasData ||
                snapshot.data!.isEmpty) {
                return const Center(

```

```

        child: Text('No Video found.'),
    );
}

return GridView.builder(
    gridDelegate:
        const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
            crossAxisSpacing: 8.0,
            mainAxisSpacing: 8.0,
        ),
    itemCount: snapshot.data!.length,
    itemBuilder: (context, index) {
        List<Music>? musics = snapshot.data;
        return Padding(
            padding: const EdgeInsets.all(10),
            child: GestureDetector(
                onTap: () {
                    controller.setIsEditVideo(
                        musics[index]);
                },
                child: Container(
                    width: 150,
                    decoration: BoxDecoration(
                        borderRadius:
                            BorderRadius.circular(
                                8.0),
                    image: DecorationImage(
                        fit: BoxFit.cover,
                        image: NetworkImage(
                            musics![index]
                                .musicCover
                                .toString(),

```

```

),
onError: (exception,
stackTrace) {
const Center(
child: Text(
"Image Error"),
);
},
),
),
padding: const EdgeInsets.all(5),
child: Column(
mainAxisAlignment:
MainAxisAlignment.end,
crossAxisAlignment:
CrossAxisAlignment
.start,
children: [
SizedBox(
width: 150,
child: Text(
musics[index]
.musicName
.toString(),
style: const TextStyle(
color: Colors.black,
fontSize: 16,
fontWeight:
FontWeight.bold,
),
maxLines: 2,
overflow: TextOverflow
.ellipsis,

```

```

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

```

manage_qna_admin_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/qa_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

```

```
class QuestionAnswerAdminScreen extends StatelessWidget {
```

```
    QuestionAnswerAdminScreen({super.key});
```

```
    final qaController = QaController();
```

```

final diagnoseController = DiagnoseController();

@Override
Widget build(BuildContext context) {
    return GetBuilder<QaController>(
        init: qaController,
        builder: (controller) {
            return WillPopScope(
                onWillPop: () async {
                    controller.onClose();
                    Get.back();
                    return true;
                },
                child: Scaffold(
                    appBar: AppBar(
                        title: const Text('Question Answer'),
                        leading: Builder(builder: (BuildContext context) {
                            return IconButton(
                                onPressed: () {
                                    controller.onClose();
                                    Get.back();
                                },
                                icon: const Icon(Icons.arrow_back));
                            }),
                    ),
                    body: Padding(
                        padding: const EdgeInsets.all(16.0),
                        child: Form(
                            key: controller.formKey.value,
                            child: Column(
                                crossAxisAlignment: CrossAxisAlignment.start,
                                children: <Widget>[

```

```

TextField(
    controller: controller.testTitleController.value,
    decoration: const InputDecoration(
        labelText: 'Symptom Code',
        border: OutlineInputBorder(),
    ),
    // initialValue: controller.testQaData['test_title'].toString(),
    validator: (value) {
        if (value == null || value.isEmpty) {
            return 'Please enter symptom code';
        }
        return null;
    },
),
const SizedBox(height: 20),
Align(
    alignment: Alignment.center,
    child: ElevatedButton(
        onPressed: () {
            controller.addTestQa();
        },
        child: const Text(
            'Add Question Answer For Certain Factor User'))),
controller.testQa.isNotEmpty
? Expanded(
    child: ListView.builder(
        itemCount: controller.testQa.length,
        itemBuilder: (context, index) {
            while (controller.formKeys.length <= index) {
                controller.formKeys
                    .add(GlobalKey<FormState>());
            }
            while (controller.formSecondKeys.length <=

```

```

index) {
controller.formSecondKeys
.add(GlobalKey<FormState>());
}

while (controller.formThirdKeys.length <=
index) {
controller.formThirdKeys
.add(GlobalKey<FormState>());
}

return Form(
key: controller.formKeys[
index], // Assign GlobalKey here
child: Column(
mainAxisAlignment:
MainAxisAlignment.start,
crossAxisAlignment:
CrossAxisAlignment.start,
children: [
Padding(
padding:
const EdgeInsets.all(8),
child: ElevatedButton(
child: const Text('Delete'),
onPressed: () {
controller
.removeTestQa(index);
}),
Padding(
padding:
const EdgeInsets.all(8),
child: Text(
'Question: ${controller.testQa[index]['question']}')),
Form(

```

```

key: controller
    .formSecondKeys[index],
child: Row(children: [
    Expanded(
        child: TextFormField(
            decoration:
                const InputDecoration(
                    labelText: 'Question',
                    border:
                        OutlineInputBorder(),
                ),
            validator: (value) {
                final question =
                    controller
                        .testQa[index]
                        ['question'];
                if (question == null ||
                    question
                        .trim()
                        .isEmpty ||
                    question == "") {
                    if (value == null ||
                        value.isEmpty) {
                        return 'Please enter question';
                    }
                    return null;
                }
                return null;
            },
            onChanged: (value) {
                controller
                    .setQuestion(value);
            },
        ),
    ],
),
);
    
```

```

        )),
        ElevatedButton(
            child: const Text('Add'),
            onPressed: () {
                if (controller
                    .formSecondKeys[
                        index]
                    .currentState!
                    .validate()) {
                    controller
                        .onSubmitQuestion(
                            index);
                    controller
                        .formSecondKeys[
                            index]
                        .currentState!
                        .reset();
                }
            })
        ])),
        Padding(
            padding:
                const EdgeInsets.all(8),
            child: Text(
                'CF expert: ${controller.testQa[index]['cf_expert']}')),
        Form(
            key: controller
                .formThirdKeys[index],
            child: Row(children: [
                Expanded(
                    child: TextFormField(
                        keyboardType:
                            TextInputType.number,

```

decoration:

```

        const InputDecoration(
      labelText:
        'Certain Factor Expert Value',
      border:
        OutlineInputBorder(),
      ),
      validator: (value) {
        if (value == null ||
            value.isEmpty) {
          return 'Please enter certain factor expert value';
        }
        return null;
      },
      onChanged: (value) {
        controller
          .setExpertAnswer(
            value);
      },
    )));
    ElevatedButton(
      child: const Text('Add'),
      onPressed: () {
        if (controller
          .formThirdKeys[
            index]
          .currentState!
          .validate()) {
          controller
            .onSubmitExpertValue(
              index);
        }
        controller
          .formThirdKeys[
    
```

```

        index]
        .currentState!
        .reset();
    }
})
]),
const Padding(
    padding: EdgeInsets.all(8),
    child: Text('Answer:')),
Row(children: [
Expanded(
    child: TextFormField(
        keyboardType:
            TextInputType.number,
        decoration:
            const InputDecoration(
                labelText: 'Key Answer',
                border: OutlineInputBorder(),
            ),
        validator: (value) {
            if (value == null ||
                value.isEmpty) {
                return 'Please enter key answer';
            }
            return null;
        },
        onChanged: (value) {
            controller
                .setKeyAnswer(value);
        },
)),
Expanded(
    child: TextFormField(

```

decoration:

```

        const InputDecoration(
        labelText: 'Name Answer',
        border: OutlineInputBorder(),
        ),
        validator: (value) {
        if (value == null ||
            value.isEmpty) {
            return 'Please enter name answer';
        }
        return null;
        },
        onChanged: (value) {
        controller
        .setNameAnswer(value);
        },
        )),
        Expanded(
        child: TextFormField(
        keyboardType:
        TextInputType.number,
        decoration:
        const InputDecoration(
        labelText:
        'Certain Factor User Value',
        border: OutlineInputBorder(),
        ),
        validator: (value) {
        if (value == null ||
            value.isEmpty) {
            return 'Please enter some certain factor user value';
        }
        return null;
        }
    
```

```

        },
        onChanged: (value) {
            controller
                .setValueAnswer(value);
        },
    )),  

    ElevatedButton(
        child: const Text('Add'),
        onPressed: () {
            if (controller
                .formKeys[index]
                .currentState!
                .validate()) {
                controller.onSubmitAnswer(
                    index);
                controller.formKeys[index]
                    .currentState!
                    .reset();
            }
        },
    )),  

    ]),  

    Column(
        mainAxisAlignment:
            MainAxisAlignment.start,
        crossAxisAlignment:
            CrossAxisAlignment.start,
        children: controller
            .testQa[index]['answer']
            .entries
            .map<Widget>((entry) {
            String key = entry.key;
            Map<String, dynamic> answer =
                entry.value;

```

```

        return Row(children: [
            Text(
                '$key: ${answer['name']} - ${answer['value']}',
                style: const TextStyle(
                    fontSize: 20)),
            ElevatedButton(
                onPressed: () =>
                    controller
                        .deleteAnswer(
                            index, key),
                child:
                    const Text('Delete'))
        ]);
    }).toList(),
)
]);
},
))
: const Align(
    alignment: Alignment.center,
    child: Padding(
        padding: EdgeInsets.all(8),
        child: Text('No Question Made Yet'))),
Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
        ElevatedButton(
            onPressed: () async {
                if (controller.formKey.value.currentState!
                    .validate()) {
                    if (controller.isEdit.value) {
                        await controller.updateQa().then((value) {

```

```

        controller.resetData());
    });
} else {
    if (controller.testQa.isEmpty) {
        Get.snackbar("Empty Field",
            "Please add atleast one Question And Answer");
    } else {
        await controller
            .insertQA()
            .then((value) {
                controller.resetData();
            });
    }
},
child: Text(controller.isEdit.value
    ? 'Update Certain Factor User'
    : 'Add Certain Factor User'),
),
ElevatedButton(
    onPressed: () => controller.resetData(),
    child: const Text('Reset Data'),
)
],
),
Padding(
    padding: const EdgeInsets.all(8.0),
    child: TextField(
        onChanged: (value) {
            controller.setSearch(value);
        },
        decoration: const InputDecoration(
            labelText: 'Search',

```

```

hintText: 'Enter search term',
prefixIcon: Icon(Icons.search),
),
),
),
),
Expanded(
child: Obx(() {
return StreamBuilder<List<Diagnose>>(
stream: controller.searchDiagnose(),
builder: (context, snapshot) {
if (snapshot.hasData) {
List<Diagnose>? diagnoses = snapshot.data!;
return ListView.builder(
itemCount: diagnoses.length,
itemBuilder: (context, index) {
return GestureDetector(
onTap: () {
controller.setEditDiagnose(
diagnoses[index]);
},
child: Card(
child: ListTile(
title: Row(
mainAxisAlignment:
MainAxisAlignment
.spaceBetween,
crossAxisAlignment:
CrossAxisAlignment
.center,
children: [
SizedBox(
width: 150,
child: Text(

```

```
    diagnoses[index].testTitle.toString(),
    style: const TextStyle(
      color: Colors.black,
      fontSize: 16,
      fontWeight: FontWeight.bold,
    ),
    maxLines: 2,
    overflow: TextOverflow.ellipsis),
  ),
),
ElevatedButton(
  child: const Text(
    'Delete Qa'),
  onPressed: () {
    if (diagnoses[index].testId != null ||
        diagnoses[index].testId != null) {
      controller.deleteQA(diagnoses[index].testId
          .toString());
    }
  },
),
]),
));
});
```

```
);

} else if (snapshot.hasError) {
    return const Center(
        child: Text(
            'Error Occurred. Please Contact Our Support Team.',
            textAlign: TextAlign.center,
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                fontSize: 16,
                fontWeight: FontWeight.w400,
                color: Colors.black,
            ),
        ),
    );
}

} else if (snapshot.connectionState ==
    ConnectionState.waiting) {
    return const Center(
        child: CircularProgressIndicator(),
    );
}

} else {
    return const Center(
        child: Text(
            'No Diagnose Data.',
            textAlign: TextAlign.center,
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                fontSize: 16,
                fontWeight: FontWeight.w400,
                color: Colors.black,
            ),
        ),
    );
}
}
```

```

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

```

manage_rule_based_admin_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/rule_based_control
ler.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/models/rule_based_diagnose.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class RuleBasedDiagnoseAdminScreen extends StatelessWidget {
    RuleBasedDiagnoseAdminScreen({super.key});

    final ruleBasedDiagnoseController = Get.put(RuleBasedDiagnoseController());

    @override
    Widget build(BuildContext context) {
        return GetBuilder<RuleBasedDiagnoseController>(
            init: ruleBasedDiagnoseController,

```

```

builder: (controller) {
  return WillPopScope(
    onWillPop: () async {
      controller.onClose();
      return true;
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text('Rule-Based Diagnose Admin'),
        leading: Builder(builder: (BuildContext context) {
          return IconButton(
            onPressed: () {
              controller.onClose();
              Get.back();
            },
            icon: const Icon(Icons.arrow_back));
        }),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Column(children: [
          Expanded(
            child: ListView(children: <Widget>[
              Form(
                key: controller.formKey.value,
                child: Column(
                  crossAxisAlignment: CrossAxisAlignment.stretch,
                  children: [
                    TextFormField(
                      validator: (value) {
                        if (value == null || value.isEmpty) {
                          return 'Please enter rule based detail';
                        }
                      }
                    )
                  ],
                ),
              ),
            ],
          ),
        ],
      ),
    ),
  );
}

```

```

        return null;
    },
    controller: controller
        .ruleBasedDetailController.value,
    decoration: const InputDecoration(
        labelText: 'Rule Based Detail'),
    maxLines: 10,
),
TextFormField(
    controller: controller.maxController.value,
    decoration: const InputDecoration(
        labelText: 'Max Percent'),
    keyboardType: TextInputType.number,
    validator: (value) {
        if (value == null || value.isEmpty) {
            return 'Please enter max percent';
        }
        return null;
    },
),
const SizedBox(height: 20),
TextFormField(
    controller: controller.minController.value,
    decoration: const InputDecoration(
        labelText: 'Min Percent'),
    keyboardType: TextInputType.number,
    validator: (value) {
        if (value == null || value.isEmpty) {
            return 'Please enter min percent';
        }
        return null;
    },
),

```

```

const SizedBox(height: 20),
TextFormField(
  controller: controller.nameController.value,
  decoration: const InputDecoration(
    labelText: 'Diagnose Name'),
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter some text';
    }
    return null;
  },
),
const SizedBox(height: 20),
ElevatedButton(
  onPressed: () async {
    if (controller.formKey.value.currentState!
        .validate()) {
      if (controller.setEdit.value) {
        await controller
          .updateRuleBased()
          .then((value) {
            controller.resetData();
          });
      } else {
        await controller
          .insertRuleBased()
          .then((value) {
            controller.resetData();
          });
      }
    }
  },
),
child: Text(controller.setEdit.value)

```

```
? 'Update Rule Based'  
: 'Add Rule Based'),  
,  
ElevatedButton(  
onPressed: () => controller.resetData(),  
child: const Text('Reset Data'),  
,  
],  
,  
),  
],  
),  
]),  
Padding(  
padding: const EdgeInsets.all(8.0),  
child: TextField(  
onChanged: (value) {  
controller.setSearch(value);  
},  
decoration: const InputDecoration(  
labelText: 'Search',  
hintText: 'Enter search term',  
prefixIcon: Icon(Icons.search),  
,  
,  
),  
),  
Expanded(  
child: StreamBuilder<List<RuleBasedDiagnose>>(  
stream: controller.searchRuleBased(),  
builder: (context, snapshot) {  
if (snapshot.connectionState ==  
ConnectionState.waiting) {  
return const Center(  
child: CircularProgressIndicator(),  
);
```

```

}

if (snapshot.hasError) {
    return Center(
        child: Text('Error: ${snapshot.error}'),
    );
}

if (!snapshot.hasData || snapshot.data!.isEmpty) {
    return const Center(
        child: Text('No Rule Based found.'),
    );
}

return GridView.builder(
    gridDelegate:
        const SliverGridDelegateWithFixedCrossAxisCount(
            crossAxisCount: 2,
            crossAxisSpacing: 8.0,
            mainAxisSpacing: 8.0,
        ),
    itemCount: snapshot.data!.length,
    itemBuilder: (context, index) {
        List<RuleBasedDiagnose>? rulebased =
            snapshot.data;
        return Padding(
            padding: const EdgeInsets.all(10),
            child: GestureDetector(
                onLongPress: () => controller
                    .deleteRuleBased(rulebased[index]
                        .ruleBasedDiagnoseId
                        .toString()),
                onTap: () {

```

```
controller  
.setEditRuleBased(rulebased[index]);  
},  
child: Container(  
width: 150,  
padding: const EdgeInsets.all(5),  
child: Column(  
mainAxisAlignment:  
MainAxisAlignment.end,  
crossAxisAlignment:  
CrossAxisAlignment.start,  
children: [  
SizedBox(  
width: 150,  
child: Text(  
rulebased![index]  

```

```

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

});

}

}

manage_video_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/video_admin_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/video_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class ManageVideoAdminScreen extends StatelessWidget {
    ManageVideoAdminScreen({super.key});
    final videoAdminController = Get.put(VideoAdminController());
    final videoController = Get.put(VideoController());

    @override
    Widget build(BuildContext context) {
        return GetBuilder<VideoAdminController>(

```

```

init: videoAdminController,
builder: (controller) {
  return WillPopScope(
    onWillPop: () async {
      controller.clearAllData();
      videoController.clearSearch();
      return true;
    },
    child: Scaffold(
      appBar: AppBar(
        title: const Text('Video'),
        leading: Builder(builder: (BuildContext context) {
          return IconButton(
            onPressed: () {
              controller.onClose();
              Get.back();
            },
            icon: const Icon(Icons.arrow_back));
          },
        ),
        body: Padding(
          padding: const EdgeInsets.all(20.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.stretch,
            children: <Widget>[
              Form(
                key: controller.formKey.value,
                child: Column(children: <Widget>[
                  TextFormField(
                    validator: (value) {
                      if (value == null || value.isEmpty) {
                        return 'Please enter some text';
                      }
                    }
                ],
              )
            ],
          ),
        ),
      ),
    ),
  );
}

```

```

        return null;
    },
    controller:
        controller.titleController.value,
    decoration: const InputDecoration(
        labelText: 'Video Title'),
),
const SizedBox(height: 20),
TextField(
    validator: (value) {
        if (value == null || value.isEmpty) {
            return 'Please enter some text';
        }
        return null;
    },
    controller:
        controller.descriptionController.value,
    decoration: const InputDecoration(
        labelText: 'Video Description'),
    maxLines: 3,
),
const SizedBox(height: 20),
SizedBox(
    width: double.infinity,
    child: ElevatedButton(
        onPressed: controller.pickImage,
        child: const Text('Pick Image'),
    )),
const SizedBox(height: 10),
controller.imageFile.value != null
? Text(controller.imageFile.value!.name)
: Container(),
const SizedBox(height: 20),

```

```

SizedBox(
    width: double.infinity,
    child: ElevatedButton(
        onPressed: controller.pickVideo,
        child: const Text('Pick Video'),
    )),
const SizedBox(height: 10),
controller.videoFile.value != null
? Text(controller.videoFile.value!.name)
: Container(),
const SizedBox(height: 20),
Row(
    mainAxisAlignment:
        MainAxisAlignment.spaceBetween,
    children: [
        ElevatedButton(
            onPressed: () async {
                if (controller
                    .formKey.value.currentState!
                    .validate() {
                    if (controller.imageFile.value !=
                        null ||
                        controller.videoFile.value !=
                        null) {
                        if (controller
                            .setEditVideo.value) {
                            await controller
                                .editVideo()
                                .then((value) {
                            controller.resetData();
                        });
                    } else {
                        await controller
                    }
                }
            }
        )
    ],
)

```

```

.uploadVideo()
.then((value) {
controller.resetData();
});
}
} else {
Get.snackbar("Empty Field",
"Image And Video must be fill");
}
},
),
child: Text(
controller.setEditable.value
?'Update'
:'Add'),
),
ElevatedButton(
onPressed: () => controller.resetData(),
child: const Text('Reset Data'),
),
],
),
],
),
],
Padding(
padding: const EdgeInsets.all(8.0),
child: TextField(
onChanged: (value) {
videoController.setSearch(value);
},
decoration: const InputDecoration(
labelText: 'Search',
hintText: 'Enter search term',
prefixIcon: Icon(Icons.search),

```

```

),
),
),
Expanded(
child: StreamBuilder<List<Video>>(
  stream: videoController.searchVideo(),
  builder: (context, snapshot) {
    if (snapshot.connectionState ==
        ConnectionState.waiting) {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }

    if (snapshot.hasError) {
      return Center(
        child: Text('Error: ${snapshot.error}'),
      );
    }

    if (!snapshot.hasData ||
        snapshot.data!.isEmpty) {
      return const Center(
        child: Text('No Video found.'),
      );
    }
  },
)

return GridView.builder(
  gridDelegate:
    const SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,
      crossAxisSpacing: 8.0,
      mainAxisSpacing: 8.0,
    )
);

```

```

),
itemCount: snapshot.data!.length,
itemBuilder: (context, index) {
  List<Video>? videos = snapshot.data;
  return Padding(
    padding: const EdgeInsets.all(10),
    child: GestureDetector(
      onTap: () {
        controller
          .setIsEditVideo(videos[index]);
      },
      child: Container(
        width: 150,
        decoration: BoxDecoration(
          borderRadius:
            BorderRadius.circular(8.0),
          image: DecorationImage(
            fit: BoxFit.cover,
            image: NetworkImage(
              videos![index]
                .videoCaptionUrl
                .toString(),
            ),
          ),
        ),
        onError:
          (exception, stackTrace) {
        const Center(
          child:
            Text("Image Error"),
        );
      },
    ),
  ),
  padding: const EdgeInsets.all(5),

```

```
child: Column(  
    mainAxisAlignment:  
        MainAxisAlignment.end,  
    crossAxisAlignment:  
        CrossAxisAlignment.start,  
    children: [  
        SizedBox(  
            width: 150,  
            child: Text(  
                videos[index]  
                    .videoTitle  
                    .toString(),  
                style: const TextStyle(  
                    color: Colors.black,  
                    fontSize: 16,  
                    fontWeight:  
                        FontWeight.bold,  
                ),  
                maxLines: 2,  
                overflow:  
                    TextOverflow.ellipsis,  
            ),  
        ),  
        ],  
    ),  
),  
);  
,  
},  
);  
,
```

```

        ]))));

    });

}

```

B) Autentikasi

email_verification_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/auth_controller.dar
t';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller
.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/sign_in_screen.dar
t';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class EmailVerificationScreen extends StatelessWidget {
    EmailVerificationScreen({super.key});

    final userGuardsController = Get.put(UserGuardsController());
    final authController = Get.put(AuthController());

    Future<void> sendEmailVerification(BuildContext context) async {
        try {
            if (userGuardsController.user.currentUser != null &&
                !userGuardsController.user.currentUser!.emailVerified) {
                await userGuardsController.user.currentUser!.sendEmailVerification();
            }
        } catch (e) {
            print(e);
        }
    }
}

```

```

Get.dialog(AlertDialog(
  title: const Text('Verification Email Sent',
    style: TextStyle(color: Colors.black)),
  content: Text(
    'A verification email has been sent to
    ${userGuardsController.user.currentUser!.email}. Please check your email to verify
    your account.'),
  actions: [
    TextButton(
      onPressed: () {
        Get.to(SignInScreen());
      },
      child: const Text('OK'),
    ),
  ],
));
}

} catch (e) {
  await authController.logout();
}
}

@Override
Widget build(BuildContext context) {
  return Scaffold(
    backgroundColor: Colors.white,
    body: Center(
      child: ElevatedButton(
        style: ElevatedButton.styleFrom(
          foregroundColor: Colors.white,
          backgroundColor: const Color.fromRGBO(5, 15, 47, 1),
          shape: RoundedRectangleBorder(

```

```

        borderRadius: BorderRadius.circular(12),
    )),
    onPressed: () async => await sendEmailVerification(context),
    child: const Text('Send Verification Email'),
),
),
);
}
}

```

forgot_password_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/sign_in_screen.dar
t';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

```

```
class ForgotPasswordScreen extends StatelessWidget {
```

```
ForgotPasswordScreen({super.key});
```

```
final TextEditingController _emailController = TextEditingController();
```

```

Future<void> _resetPassword() async {
try {
if (_emailController.text.isNotEmpty) {
await FirebaseAuth.instance
.sendPasswordResetEmail(email: _emailController.text);
Get.dialog(AlertDialog(
title: const Text('Password Reset Email Sent'),
content: const Text(

```

```

'Check your email for instructions to reset your password.'),
actions: [
TextButton(
 onPressed: () {
Get.to(SignInScreen());
},
child: const Text('OK'),
),
],
));
} else {
Get.dialog(const Text('Please fill your email'));
}
} catch (e) {
Get.dialog(AlertDialog(
title: const Text('Error'),
content: Text(e.toString()),
actions: [
TextButton(
 onPressed: () {
Get.back();
},
child: const Text('OK'),
),
],
));
}
}

@Override
Widget build(BuildContext context) {
return Scaffold(
backgroundColor: Colors.white,

```

```

body: Padding(
  padding: const EdgeInsets.all(16.0),
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      TextField(
        controller: _emailController,
        decoration: const InputDecoration(
          labelText: 'Email',
        ),
      ),
      const SizedBox(height: 20),
      ElevatedButton(
        style: ElevatedButton.styleFrom(
          foregroundColor: Colors.white,
          backgroundColor: const Color.fromRGBO(5, 15, 47, 1),
          shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.circular(12),
          )),
        onPressed: _resetPassword,
        child: const Text('Reset Password'),
      ),
    ],
  ),
);
}
}

```

sign_in_screen.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/auth_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/forgot_password_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/sign_up_screen.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class SignInScreen extends StatelessWidget {
  static const String id = "sign_in_screen";

  final authController = Get.put(AuthController());

  final _formKey = GlobalKey<FormState>();

  SignInScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Container(
        decoration: const BoxDecoration(
          color: Colors.white,
        ),
        child: ListView(
          children: <Widget>[
            Image.asset('assets/images/people.jpg', width: 300, height: 300),
            const Align(
              alignment: Alignment.centerLeft,

```

```

child: Padding(
    padding: EdgeInsets.all(20),
    child: Text('Sign In',
        style: TextStyle(
            fontFamily: 'MochiyPopOne',
            fontSize: 32,
            fontWeight: FontWeight.w400,
            color: Colors.white,
        ))),
Form(
    key: _formKey,
    child: Column(children: [
        Padding(
            padding: const EdgeInsets.all(20),
            child: TextFormField(
                controller: authController.email,
                keyboardType: TextInputType.emailAddress,
                validator: (value) {
                    if (value == null || value.isEmpty) {
                        return 'please enter your email address';
                    }
                    return null;
                },
                decoration: InputDecoration(
                    labelText: 'Email',
                    filled: true,
                    fillColor: Colors.white,
                    border: OutlineInputBorder(
                        borderRadius: BorderRadius.circular(30),
                    ),
                ),
            )),
        Padding(
    
```

```

padding: const EdgeInsets.all(20),
child: TextFormField(
  controller: authController.password,
  keyboardType: TextInputType.visiblePassword,
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'please enter your password';
    }
    return null;
  },
  decoration: InputDecoration(
    labelText: 'Password',
    filled: true,
    fillColor: Colors.white,
    border: OutlineInputBorder(
      borderRadius: BorderRadius.circular(30),
    ),
  ),
  obscureText: true,
)),
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 20),
  child: Row(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      GestureDetector(
        onTap: () async {
          Get.to(ForgotPasswordScreen());
        },
      ),
      child: const Text(
        'Forgot Password?',
        style: TextStyle(
          fontFamily: 'MochiyPopOne',

```

```

        color: Colors.black,
        decoration: TextDecoration.underline,
      ),
      ),
      ],
      ),
      ),
      ),
      Padding(
        padding: const EdgeInsets.all(20),
        child: ElevatedButton(
          onPressed: () async {
            if (_formKey.currentState!.validate()) {
              await authController.login();
            }
          },
          style: ElevatedButton.styleFrom(
            foregroundColor: Colors.black,
            backgroundColor: const Color.fromRGBO(5, 15, 47, 1),
            minimumSize: const Size.fromHeight(50),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(30),
            ),
            ),
            ),
            child: const Text('Sign In',
            style: TextStyle(
              fontFamily: 'MochiyPopOne', color: Colors.white)),
            )),
            Padding(
            padding: const EdgeInsets.all(20),
            child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [

```

```
const Text('Need to create an account? ',  
        style: TextStyle(  
            fontFamily: 'MochiyPopOne',  
            color: Colors.black)),  
        ElevatedButton(  
            style: ElevatedButton.styleFrom(  
                foregroundColor: Colors.white,  
                backgroundColor: const Color.fromRGBO(5, 15, 47, 1),  
                shape: RoundedRectangleBorder(  
                    borderRadius: BorderRadius.circular(12),  
                )),  
            onPressed: () => Get.to(SignUpScreen()),  
            child: const Text('Sign Up',  
                style: TextStyle(  
                    fontFamily: 'MochiyPopOne',  
                    color: Colors.white,  
                    fontWeight: FontWeight.bold)),  
        ),  
    ],  
),  
),  
],  
),  
);  
}  
}
```

sign_up_screen.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/auth_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/sign_in_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/auth_screen/email_verification_screen.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class SignUpScreen extends StatelessWidget {
    static const String id = "sign_up_screen";

    final _formKey = GlobalKey<FormState>();

    final signUpController = Get.put(AuthController());

    SignUpScreen({super.key});

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Container(
                decoration: const BoxDecoration(
                    color: Colors.white,
                ),
                child: ListView(children: <Widget>[
                    Image.asset('assets/images/people.jpg', width: 300, height: 300),
                    const Align(
                        alignment: Alignment.centerLeft,
                        child: Padding(

```

```

padding: EdgeInsets.all(10),
child: Text('Sign Up',
style: TextStyle(
fontFamily: 'MochiyPopOne',
fontSize: 32,
fontWeight: FontWeight.w400,
color: Colors.white,
))),

Form(
key: _formKey,
child: Column(children: [
Padding(
padding: const EdgeInsets.all(10),
child: TextFormField(
validator: (value) {
if (value == null || value.isEmpty) {
return 'please enter your username';
}
return null;
},
controller: signUpController.username,
decoration: InputDecoration(
labelText: 'Username',
filled: true,
fillColor: Colors.white,
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(30),
),
),
),
)),
Padding(
padding: const EdgeInsets.all(10),
child: TextFormField(

```

```

validator: (value) {
  if (value == null || value.isEmpty) {
    return 'please enter your email';
  }
  return null;
},
controller: signUpController.email,
keyboardType: TextInputType.emailAddress,
decoration: InputDecoration(
  labelText: 'Email',
  filled: true,
  fillColor: Colors.white,
  border: OutlineInputBorder(
    borderRadius: BorderRadius.circular(30),
  ),
),
),
Padding(
  padding: const EdgeInsets.all(10),
  child: TextFormField(
    validator: (value) {
      if (value == null || value.isEmpty) {
        return 'please enter your password';
      }
      return null;
    },
    controller: signUpController.password,
    decoration: InputDecoration(
      labelText: 'Password',
      filled: true,
      fillColor: Colors.white,
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(30),
      ),
    ),
  ),
);

```

```

),
),
obscureText: true,
)),
Padding(
padding: const EdgeInsets.all(10),
child: TextFormField(
validator: (value) {
if (value == null || value.isEmpty) {
return 'please enter your confirm password';
}
if (value != signUpController.password.text) {
return 'confirm password does not match';
}
return null;
},
controller: signUpController.confirmPassword,
decoration: InputDecoration(
labelText: 'Confirm Password',
filled: true,
fillColor: Colors.white,
border: OutlineInputBorder(
borderRadius: BorderRadius.circular(30),
),
),
),
obscureText: true,
)),
Padding(
padding: const EdgeInsets.all(10),
child: ElevatedButton(
onPressed: () async {
if (_formKey.currentState!.validate()) {
await signUpController.register().then((value) =>

```



```

        onPressed: () => Get.to(SignInScreen()),
        child: const Text('Sign In',
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                color: Colors.white,
                fontWeight: FontWeight.bold)),
        ),
    )));
}),
]);
}
}

```

C) User

```

breathing_exercise_screen.dart
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/breathing_exercise
_controller.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class BreatheExcerciseScreen extends StatelessWidget {
    final breathingExerciseController = Get.put(BreathingExerciseController());

```

```
BreatheExcerciseScreen({super.key});
```

```

Widget actionButton(
    SessionState state, BreathingExerciseController controller) {
    if (state == SessionState.initial) {
        return Padding(
            padding: const EdgeInsets.only(bottom: 8.0),
            child: FloatingActionButton(

```

```

        backgroundColor: Colors.yellow.shade900,
        onPressed: controller.start,
        child: const Icon(Icons.play_arrow),
      ),
    );
} else if (state == SessionState.ended) {
  return FloatingActionButton(
    backgroundColor: Colors.yellow.shade900,
    onPressed: () {
      controller.resetAll();
    },
    child: const Icon(Icons.arrow_back));
} else {
  return FloatingActionButton(
    backgroundColor: Colors.yellow.shade900,
    onPressed: controller.stop,
    child: const Icon(Icons.stop));
}
}

@Override
Widget build(BuildContext context) {
  return GetX<BreathingExerciseController>(
    init: breathingExerciseController,
    builder: (controller) {
      String minutesStr =
        (controller.minutes.value % 60).toString().padLeft(2, '0');
      String secondsStr =
        (controller.seconds.value % 60).toString().padLeft(2, '0');
      return Scaffold(
        backgroundColor: const Color.fromRGBO(255, 253, 208, 1),
        body: Align(
          alignment: Alignment.center,

```

```

child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
        Row(mainAxisAlignment: MainAxisAlignment.center, children: [
            RepaintBoundary(
                child: SizedBox(
                    height: 90.0,
                    child: Text('$minutesStr:$secondsStr',
                        style: controller
                            .dependencies.value.textStyle.value),
                ),
            )
        ]),
        Padding(
            padding: const EdgeInsets.only(top: 15),
            child: Text(
                controller
                    .instructionText(controller.sessionState.value),
                style: const TextStyle(fontSize: 28),
            ),
        ),
        Padding(
            padding: const EdgeInsets.only(top: 15),
            child: Text(
                '${controller.countDown.value}',
                style: const TextStyle(
                    fontWeight: FontWeight.bold, fontSize: 28),
            )));
    ],
),
floatingActionButton:
    actionButton(controller.sessionState.value, controller),
floatingActionButtonLocation: FloatingActionButtonLocation.endFloat,

```

```
 );  
});  
}  
}
```

crisis_support_screen.dart

```
import  
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/crisis_support_controller.dart';  
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/crisis_support.dart';  
import 'package:flutter/material.dart';  
import 'package:flutter_phone_direct_caller/flutter_phone_direct_caller.dart';  
import 'package:flutter_svg/flutter_svg.dart';  
import 'package:get/get.dart';  
  
class CrisisSupportScreen extends StatelessWidget {  
    static const String id = "crisis_support_screen";  
  
    final crisisSupportController =  
        Get.put(CrisisSupportController());  
  
    CrisisSupportScreen({super.key});  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            body: StreamBuilder<List<CrisisSupport>>(  
                stream: crisisSupportController.getAllCrisisSupport(),  
                builder: (context, snapshot) {  
                    List<CrisisSupport>? crisisSupportData = snapshot.data;  
                    if (snapshot.hasData) {  
                        return Container(  
                            child:  
                                ...  
                        );  
                    } else {  
                        return Center(child: CircularProgressIndicator());  
                    }  
                },  
            ),  
        );  
    }  
}
```

```

decoration: const BoxDecoration(
  gradient: LinearGradient(
    begin: Alignment.topRight,
    end: Alignment.bottomLeft,
    colors: [
      Color.fromRGBO(255, 253, 208, 1),
      Color.fromRGBO(255, 255, 255, 1),
    ]),
),
child: ListView.builder(
  itemCount: crisisSupportData?.length,
  itemBuilder: (context, index) {
    return Container(
      padding: const EdgeInsets.all(10),
      margin: const EdgeInsets.all(10),
      width: double.infinity,
      decoration: BoxDecoration(
        color: const Color.fromRGBO(255, 220, 220, 1),
        borderRadius: BorderRadius.circular(20.0)),
      child: Column(
        children: [
          Padding(
            padding: const EdgeInsets.all(10),
            child: Text(
              'Nama Rumah Sakit:
${crisisSupportData![index].hospitalName.toString()}',
              style: const TextStyle(
                fontFamily: 'MochiyPopOne',
                fontSize: 5,
                color: Colors.black))),
          Padding(
            padding: const EdgeInsets.all(10),
            child: Text(

```

```

'Phone Number:  

${crisisSupportData[index].hospitalContact.toString()},  

style: const TextStyle(  

    fontFamily: 'MochiyPopOne',  

    fontSize: 5,  

    color: Colors.black))),  

Padding(  

    padding: const EdgeInsets.all(10),  

    child: Text(  

        overflow: TextOverflow.ellipsis,  

        'Alamat:  

${crisisSupportData[index].hospitalAddress.toString()},  

style: const TextStyle(  

    fontFamily: 'MochiyPopOne',  

    fontSize: 5,  

    color: Colors.black)),  

),  

Padding(  

    padding: const EdgeInsets.all(10),  

    child: IconButton(  

        icon: SvgPicture.asset(  

            'assets/svg/phone-call.svg'),  

        onPressed: () async {  

            await FlutterPhoneDirectCaller  

                .callNumber(  

                    crisisSupportData[index]  

                        .hospitalContact  

                        .toString());  

            })))  

],  

));
},  

),

```

```
    );  
  } else if (snapshot.hasError) {  
    return Center(  
      child: Text(snapshot.error.toString()),  
    );  
  } else {  
    return const Center(  
      child: CircularProgressIndicator(),  
    );  
  }  
});  
}  
}
```

diagnose_history_screen.dart

```
import 'dart:convert';

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_history_
controller.dart';

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

class DiagnoseHistoryScreen extends StatelessWidget {
    static const String id = "diagnose_history_screen";

    final diagnoseHistoryController = Get.put(DiagnoseHistoryController());

    DiagnoseHistoryScreen({super.key});
```

```

return Scaffold(
  appBar: AppBar(
    backgroundColor: const Color.fromRGBO(255, 253, 208, 1),
    title:
      const Text('Diagnose History', style: TextStyle(color: Colors.black)),
    centerTitle: true,
    iconTheme: const IconThemeData(color: Colors.black),
  ),
  body: StreamBuilder<List<Map<String, dynamic>>>(
    stream: diagnoseHistoryController.getHistoryDiagnoseByUid(),
    initialData: const [],
    builder: (context, snapshot) {
      List<Map<String, dynamic>>? diagnose = snapshot.data!;
      if (snapshot.hasData && snapshot.data != null) {
        return Container(
          decoration: const BoxDecoration(
            gradient: LinearGradient(
              begin: Alignment.topRight,
              end: Alignment.bottomLeft,
              colors: [
                Color.fromRGBO(255, 253, 208, 1),
                Color.fromRGBO(255, 255, 255, 1),
              ],
            )),
          child: ListView.builder(
            itemCount: diagnose.length,
            itemBuilder: (context, index) {
              return Container(
                padding: const EdgeInsets.all(10),
                margin: const EdgeInsets.all(10),
                width: double.infinity,
                decoration: BoxDecoration(
                  color: const Color.fromRGBO(255, 220, 220, 1),
                  borderRadius: BorderRadius.circular(20.0)),

```

```

child: Column(children: [
  const Center(
    child: Text(
      "Result Mental Health Report:",
      textAlign: TextAlign.center,
      style: TextStyle(
        fontFamily: 'MochiyPopOne',
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: Colors.black,
      )));
  Center(
    child: Text(
      "Your Level Of Depression:
      ${diagnose[index]["diagnose_history_level_depression"]}%
      .toString(),
      textAlign: TextAlign.center,
      style: const TextStyle(
        fontFamily: 'MochiyPopOne',
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: Colors.black,
      )));
  const Divider(),
  Column(
    children: diagnose[index]
      ["diagnose_history_user_answer"]
      .entries
      .map<Widget>((MapEntry<dynamic, dynamic> e) {
        var name = json
          .decode(e.value)["name"]
          .toString();
        return ListTile(

```

```

        title: Text('Question: ${e.key.toString()}'),
        subtitle: Text('Answer: $name'),
      );
    }).toList(),
),
Center(
  child: Text(
    "Types of Symptoms:  

${diagnose[index]['diagnose_history_name']}  

    .toString(),
    textAlign: TextAlign.center,
    style: const TextStyle(
      fontFamily: 'MochiyPopOne',
      fontSize: 12,
      fontWeight: FontWeight.w400,
      color: Colors.black,
    )),
  const Divider(),
  Center(
    child: Text(
      "Detail Symptoms and recommendations"  

    .toString(),
    textAlign: TextAlign.center,
    style: const TextStyle(
      fontFamily: 'MochiyPopOne',
      fontSize: 12,
      fontWeight: FontWeight.w400,
      color: Colors.black,
    )),
  Center(
    child: Text(diagnose[index]
      ['diagnose_history_detail']
    .toString())))
  
```

```
        ]));
    },
),
);
} else if (snapshot.hasError) {
    return Center(
        child: Text(snapshot.error.toString()),
    );
} else {
    return const Center(
        child: CircularProgressIndicator(),
    );
}
)));
}
}
```

diagnose_result_screen.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_history_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/rule_based_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose_history.dart';
```

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/landing_screen.dart';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'dart:convert';

class DiagnoseResultScreen extends StatelessWidget {
DiagnoseResultScreen({super.key});

final ruleBasedDiagnoseController = Get.put(RuleBasedDiagnoseController());

final diagnoseController = Get.put(DiagnoseController());

final diagnoseHistoryController = Get.put(DiagnoseHistoryController());

final userGuardsController = Get.put(UserGuardsController());

@Override
Widget build(BuildContext context) {
    return GetBuilder<RuleBasedDiagnoseController>(
        init: ruleBasedDiagnoseController,
        builder: (controller) {
            return WillPopScope(
                onWillPop: () async {
                    Get.to(LandingScreen());
                    diagnoseController.abortAllSelectionOptions();
                    return true;
                },
                child: Scaffold(
                    body: Container(
                        decoration: const BoxDecoration(
```

```

gradient: LinearGradient(
    begin: Alignment.topCenter,
    end: Alignment.bottomCenter,
    colors: [
        Color.fromRGBO(255, 253, 208, 1),
        Color.fromRGBO(255, 255, 255, 1),
    ]),
child: Align(
    alignment: Alignment.center,
    child: StreamBuilder(
        stream: controller.getRuleBasedDiagnose(),
        builder: (context, snapshot) {
            if (snapshot.hasData && snapshot.data != null) {
                final diagnose = snapshot.data!;
                return ListView(children: [
                    Center(
                        child: Text(
                            "Result Mental Health Report: "
                            .toString(),
                            textAlign: TextAlign.center,
                            style: const TextStyle(
                                fontFamily: 'MochiyPopOne',
                                fontSize: 12,
                                fontWeight: FontWeight.w400,
                                color: Colors.black,
                            )));
                ],
            }
        }
    ),
    Center(
        child: Text(
            "Your Level Of Depression:
            ${diagnoseController.resultTest.value.round()}%"
            .toString(),
            textAlign: TextAlign.center,
            style: const TextStyle(

```

```

        fontFamily: 'MochiyPopOne',
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: Colors.black,
      )),
    const Divider(),
    Column(
      children: diagnoseController
        .mapSelectedOptions.entries
        .map((e) {
      var name = json
        .decode(e.value)["name"]
        .toString();
      return ListTile(
        title: Text(
          'Question: ${e.key.toString()}'),
        subtitle: Text('Answer: $name'));
    }).toList(),
    Center(
      child: Text(
        "Types of Symptoms:
${diagnose['rule_based_diagnose_name']}"

        .toString(),
      textAlign: TextAlign.center,
      style: const TextStyle(
        fontFamily: 'MochiyPopOne',
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: Colors.black,
      )),
    const Divider(),
    Center(
      child: Text(

```

```

    "Detail Symptoms and recommendations"
        .toString(),
    textAlign: TextAlign.center,
    style: const TextStyle(
        fontFamily: 'MochiyPopOne',
        fontSize: 12,
        fontWeight: FontWeight.w400,
        color: Colors.black,
    )),
    Center(
        child: Text(diagnose[
            'rule_based_diagnose_detail']
        .toString())),
    const Divider(),
    Center(
        child: ElevatedButton(
            onPressed: () async => await diagnoseHistoryController
                .insertToDiagnoseHistory(DiagnoseHistory(
                    diagnoseHistoryDate:
                        Timestamp.now(),
                    diagnoseHistoryRulebasedId:
                        diagnose[
                            'rule_based_diagnose_id'],
                    diagnoseHistoryLevelDepression:
                        diagnoseController.resultTest.value.round(),
                    diagnoseHistoryUserAnswer:
                        diagnoseController
                            .mapSelectedOptions,
                    userId:
                        userGuardsController
                            .user
                            .currentUser!
                            .uid)),
    
```

```

        child: const Text('Save Diagnose',
                      style: TextStyle(
                        fontFamily:
                          'MochiyPopOne',
                        fontSize: 10,
                        fontWeight:
                          FontWeight.w400,
                        color: Colors.black,
                      ))),
      );
    } else if (snapshot.hasError) {
      return Center(
        child: Text(
          'Error Occured Please Contact Our Support Team:
${snapshot.error.toString()}',
          textAlign: TextAlign.center,
          style: const TextStyle(
            fontFamily: 'MochiyPopOne',
            fontSize: 32,
            fontWeight: FontWeight.w400,
            color: Colors.black,
          )),
    );
  } else if (snapshot.connectionState ==
    ConnectionState.waiting) {
    return const Center(
      child: CircularProgressIndicator(),
    );
  } else if (snapshot.data == null) {
    return const Center(
      child: Text('No Diagnose Data',
                  textAlign: TextAlign.center,
                  style: TextStyle(

```

```

        fontFamily: 'MochiyPopOne',
        fontSize: 32,
        fontWeight: FontWeight.w400,
        color: Colors.black,
      )));
    } else {
      return const Center(
        child: CircularProgressIndicator(),
      );
    }
  })));
}

}
}

```

diagnose_screen.dart

```

import 'dart:convert';

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/diagnose_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/diagnose.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/diagnose_result_screen.dart';
import 'package:flutter/material.dart';
import 'package:flutter_swiper_view/flutter_swiper_view.dart';
import 'package:get/get.dart';

class DiagnoseScreen extends StatelessWidget {
  DiagnoseScreen({super.key});
  static const String id = "diagnose_screen";

```

```

final diagnoseController = Get.put(DiagnoseController());

final swiperController = SwiperController();

List<Widget> createWidgetList(
    DiagnoseController controller, List<dynamic>? data) {
  return data!.map((val) {
    final question = val['question'];

    final answerMap = val['answer'] as Map<String, dynamic>;

    List<Widget> radioListTiles = answerMap.entries.map((entry) {
      final answerValue = entry.value;
      return RadioListTile(
        title: Text(answerValue['name'].toString()),
        style: const TextStyle(
          fontFamily: 'MochiyPopOne',
          fontSize: 10,
          fontWeight: FontWeight.w400,
          color: Colors.black,
        ),
        value: jsonEncode(answerValue),
        groupValue: controller.mapSelectedOptions[question],
        onChanged: (dynamic value) {
          controller.setSelectionOptions(question, value);
        },
      );
    }).toList();

    return Column(children: [
      ListTile(
        title: Text(question.toString()),

```

```

textAlign: TextAlign.left,
style: const TextStyle(
    fontFamily: 'MochiyPopOne',
    fontSize: 10,
    fontWeight: FontWeight.w400,
    color: Colors.black,
)),
)...radioListTiles,
controller.isRadioButtonNull(question),
const Divider()
]);
}).toList();
}

@Override
Widget build(BuildContext context) {
return GetBuilder<DiagnoseController>(
    init: diagnoseController,
    builder: (controller) {
        return WillPopScope(
            onWillPop: () async {
                controller.abortAllSelectionOptions();
                return true;
            },
            child: Scaffold(
                body: Container(
                    decoration: const BoxDecoration(
                        gradient: LinearGradient(
                            begin: Alignment.topCenter,
                            end: Alignment.bottomCenter,
                            colors: [
                                Color.fromRGBO(208, 207, 184, 1),
                                Color.fromRGBO(255, 255, 255, 1),

```

```

]),

child: StreamBuilder<List<Diagnose>>(
    stream: controller.getAllDiagnose(),
    builder: (context, snapshot) {
        if (snapshot.hasData) {
            List<Diagnose>? diagnose = snapshot.data!;

            final questionsList = diagnose
                .map((data) => (data.testQa)
                    .map((qaItem) =>
                        qaItem['question'].toString())
                    .toList())
                .expand((questions) => questions)
                .toList();

            controller.setQuestions(questionsList);

            return Swiper(
                controller: swiperController,
                onIndexChanged: (int index) {
                    if ((diagnose.length - 1) == index) {
                        controller.setShowSubmitButton(true);
                    } else {
                        controller.setShowSubmitButton(false);
                    }
                },
                itemBuilder: (context, index) {
                    final testData = diagnose[index].testQa;

                    return ListView(children: [
                        ...createWidgetList(controller, testData),
                        controller.showSubmitButton.value
                            ? Row(

```

```

mainAxisAlignment:
    MainAxisAlignment.spaceEvenly,
children: [
    ElevatedButton(
        onPressed: () {
            swiperController
                .previous();
        },
        child: const Text('Prev',
            style: TextStyle(
                fontFamily:
                    'MochiyPopOne',
                fontSize: 10,
                fontWeight:
                    FontWeight.w400,
                color: Colors.black,
            )));
    ElevatedButton(
        onPressed: () async {
            if (controller
                .allOptionsSelected()) {
                await controller
                    .calculateDiagnose();
                Get.offAll(
                    DiagnoseResultScreen());
            } else {
                Get.snackbar(
                    "Empty Field",
                    "Check Again Your Answer");
            }
        },
        child: const Text(
            'Submit',

```

```

        style: TextStyle(
            fontFamily:
                'MochiyPopOne',
            fontSize: 10,
            fontWeight:
                FontWeight.w400,
            color: Colors.black,
        )))
    ])
: Row(
    mainAxisAlignment: index == 0
        ? MainAxisAlignment.center
        : MainAxisAlignment
            .spaceEvenly,
    children: [
        index == 0
            ? const SizedBox.shrink()
            : ElevatedButton(
                onPressed: () {
                    swiperController
                        .previous();
                },
                child: const Text(
                    'Prev',
                    style: TextStyle(
                        fontFamily:
                            'MochiyPopOne',
                        fontSize: 10,
                        fontWeight:
                            FontWeight
                                .w400,
                        color: Colors
                            .black,
                ),
            ),
    ],
)

```

```

        ))),
        ElevatedButton(
            onPressed: () {
                swiperController.next();
            },
            child: const Text('Next',
                style: TextStyle(
                    fontFamily:
                        'MochiyPopOne',
                    fontSize: 10,
                    fontWeight:
                        FontWeight.w400,
                    color: Colors.black,
                )));
        )));
    );
},
loop: false,
itemCount: diagnose.length,
physics: const NeverScrollableScrollPhysics(),
pagination: null,
control: null);
} else if (snapshot.hasError) {
    return const Center(
        child: Text(
            'Error Occured Please Contact Out Support Team',
            textAlign: TextAlign.center,
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                fontSize: 32,
                fontWeight: FontWeight.w400,
                color: Colors.black,
            )));
}

```

```
    );  
  } else if (snapshot.connectionState ==  
    ConnectionState.waiting) {  
    return const Center(  
      child: CircularProgressIndicator(),  
    );  
  } else if (snapshot.data == null) {  
    return const Center(  
      child: Text('No Diagnose Data',  
        textAlign: TextAlign.center,  
        style: TextStyle(  
          fontFamily: 'MochiyPopOne',  
          fontSize: 32,  
          fontWeight: FontWeight.w400,  
          color: Colors.black,  
        )),  
  } else {  
    return const Center(  
      child: CircularProgressIndicator(),  
    );  
  }  
});  
});  
}  
}
```

landing_screen.dart

```
import 'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/audio_controller.dart';
```

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/video_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/breathing_exercise_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/diagnose_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/search_music_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/search_video_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/setting_screen.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class LandingScreen extends StatelessWidget {
  LandingScreen({super.key});
  static const String id = "landing_screen";

  final userGuardsController = Get.put(UserGuardsController());
  final videoController = Get.put(VideoController());
  final musicController = Get.put(MusicController());
```

```

@Override
Widget build(BuildContext context) {
    return Scaffold(
        body: Container(
            decoration: const BoxDecoration(
                gradient: LinearGradient(
                    begin: Alignment.topCenter,
                    end: Alignment.bottomCenter,
                    colors: [
                        Color.fromRGBO(255, 253, 208, 1),
                        Color.fromRGBO(255, 255, 255, 1),
                    ]),
            ),
            child: ListView(children: <Widget>[
                const SizedBox(height: 10),
                Align(
                    alignment: Alignment.topCenter,
                    child: Container(
                        margin: const EdgeInsets.all(8),
                        padding: const EdgeInsets.all(8),
                        decoration:
                            BoxDecoration(borderRadius: BorderRadius.circular(10.0)),
                        child: Row(
                            mainAxisAlignment: MainAxisAlignment.spaceBetween,
                            children: [
                                Container(
                                    alignment: Alignment.center,
                                    child: Column(
                                        mainAxisSizeAlignment: MainAxisSizeAlignment.center,
                                        crossAxisAlignment: CrossAxisAlignmentAlignment.start,
                                        children: [
                                            const Text('Hi'),
                                        ],
                                    ),
                                ),
                            ],
                        ),
                    ),
                ),
            ],
        ),
    );
}

```

```

style: TextStyle(
    fontFamily: 'MochiyPopOne',
    fontWeight: FontWeight.bold,
    fontSize: 20,
    color: Color.fromRGBO(21, 0, 67, 1))),
Text(
    userGuardsController.user.currentUser!.displayName ??
"Unknown",
    style: const TextStyle(
        fontFamily: 'MochiyPopOne',
        fontWeight: FontWeight.bold,
        fontSize: 20,
        color: Color.fromRGBO(21, 0, 67, 1)),
    const Text("Let's manage your mental ",
        style: TextStyle(
            fontFamily: 'MochiyPopOne',
            color: Color.fromRGBO(21, 0, 67, 1)),
        const Text('with us!',
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                color: Color.fromRGBO(21, 0, 67, 1)),
            ],
        ),
    ),
),
],
),
),
Align(
    alignment: Alignment.center,
    child: Column(
        children: [
            Container(
                alignment: Alignment.topCenter,

```

```

margin: const EdgeInsets.all(10),
padding: const EdgeInsets.all(10),
child: Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
        const Text('Video Learning',
            style: TextStyle(
                fontFamily: 'OdorMeanChey',
                fontWeight: FontWeight.bold)),
        TextButton(
            onPressed: () => Get.to(const SearchVideoScreen()),
            child: const Text(
                'See All',
                style: TextStyle(
                    fontFamily: 'OdorMeanChey',
                    fontWeight: FontWeight.bold,
                    color: Color.fromRGBO(221, 56, 56, 1))),
        )))
    ],
),
]),
),
),
),
SizedBox(
height: 200.0,
child: Padding(
padding: const EdgeInsets.all(10),
child: StreamBuilder<List<Video>>(
stream: videoController.getAllVideo(),
builder: (context, snapshot) {
if (snapshot.hasData) {
List<Video>? videos = snapshot.data;
return ListView.builder(
scrollDirection: Axis.horizontal,

```

```
itemCount: videos!.length,  
itemBuilder: (context, index) {  
    return Padding(  
        padding: const EdgeInsets.all(10),  
        child: GestureDetector(  
            onTap: () => videoController.setVideo(videos[index]),  
            child: Container(  
                width: 150,  
                decoration: BoxDecoration(  
                    borderRadius: BorderRadius.circular(8.0),  
                    image: DecorationImage(  
                        fit: BoxFit.cover,  
                        image: NetworkImage(  
                            videos[index]  
                                .videoCaptionUrl  
                                .toString(),  
                            ),  
                        onError: (exception, stackTrace) {  
                            // Tampilkan teks jika gambar gagal dimuat  
                            const Center(  
                                child: Text("Image Error"),  
                            );  
                        },  
                    ),  
                ),  
                padding: const EdgeInsets.all(5),  
                child: Column(  
                    mainAxisAlignment: MainAxisAlignment.end,  
                    crossAxisAlignment:  
                        CrossAxisAlignment.start,  
                    children: [  
                        SizedBox(  
                            width: 150,
```

```
        child: Text(  
            videos[index].videoTitle.toString(),  
            style: const TextStyle(  
                color: Colors.black,  
                fontSize: 16,  
                fontWeight: FontWeight.bold,  
            ),  
            maxLines: 2,  
            overflow: TextOverflow.ellipsis,  
        ),  
        ),  
    ],  
),  
(  
),  
),  
);  
},  
);  
}  
} else if (snapshot.hasError) {  
    return Center(  
        child: Text(snapshot.error.toString()),  
    );  
} else {  
    return const Center(  
        child: CircularProgressIndicator(),  
    );  
}  
},  
),  
),  
Align(  
    alignment: Alignment.center,
```

```

child: Column(
  children: [
    Container(
      alignment: Alignment.topCenter,
      margin: const EdgeInsets.all(10),
      padding: const EdgeInsets.all(10),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          const Text('Relaxing Music',
            style: TextStyle(
              fontFamily: 'OdorMeanChey',
              fontWeight: FontWeight.bold)),
          TextButton(
            onPressed: () => Get.to(const SearchMusicScreen()),
            child: const Text(
              'See All',
              style: TextStyle(
                fontFamily: 'OdorMeanChey',
                fontWeight: FontWeight.bold,
                color: Color.fromRGBO(221, 56, 56, 1)),
            )))
        ],
      ),
    ),
  ],
),
SizedBox(
  height: 200.0,
  child: Padding(
    padding: const EdgeInsets.all(10),
    child: StreamBuilder<List<Music>>(
      stream: musicController.getAllMusic(),
      builder: (context, snapshot) {

```

```

List<Music>? music = snapshot.data;
if (snapshot.hasData) {
    return ListView.builder(
        scrollDirection: Axis.horizontal,
        itemCount: music!.length,
        itemBuilder: (context, index) {
            return Padding(
                padding: const EdgeInsets.all(10),
                child: Container(
                    width: 150,
                    decoration: BoxDecoration(
                        image: DecorationImage(
                            fit: BoxFit.cover,
                            image: NetworkImage(music[index]
                                .musicCover
                                .toString())),
                    ),
                    onError: (exception, stackTrace) {
                        Center(
                            child: Image.asset(
                                "assets/images/default-cover-music.jpg"),
                        );
                    },
                ),
                padding: const EdgeInsets.all(5),
                child: GestureDetector(
                    onTap: () {
                        musicController.setAudio(music[index]);
                    },
                    child: Text(music[index]
                        .musicName
                        .toString(),
                )));
}

```

```

    });
} else if (snapshot.hasError) {
    return Center(
        child: Text(snapshot.error.toString()),
    );
} else {
    return const Center(
        child: CircularProgressIndicator(),
    );
},
),
),
GestureDetector(
    onTap: () => Get.to(DiagnoseScreen()),
    child: Align(
        alignment: Alignment.center,
        child: Container(
            height: 250,
            decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(10.0),
                image: const DecorationImage(
                    fit: BoxFit.cover,
                    image: AssetImage('assets/gif/heart.gif'),
                )),
            alignment: Alignment.topLeft,
            margin: const EdgeInsets.all(10),
            padding: const EdgeInsets.all(10),
            child: const Text('Diagnose',
                style: TextStyle(
                    fontFamily: 'OdorMeanChey',
                    fontWeight: FontWeight.bold))),
),
),

```

```

),
GestureDetector(
    onTap: () => Get.to(BreatheExcerciseScreen()),
    child: Align(
        alignment: Alignment.center,
        child: Container(
            height: 250,
            alignment: Alignment.topLeft,
            decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(10.0),
                image: const DecorationImage(
                    fit: BoxFit.cover,
                    image:
                        AssetImage('assets/gif/breathing_exercise.gif'),
                )),
            margin: const EdgeInsets.all(10),
            padding: const EdgeInsets.all(10),
            child: const Text('Breathing Exercise',
                style: TextStyle(
                    fontFamily: 'OdorMeanChey',
                    fontWeight: FontWeight.bold)),
        ),
    )));
GestureDetector(
    onTap: () => Get.to(SettingScreen()),
    child: Align(
        alignment: Alignment.center,
        child: Container(
            height: 250,
            decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(10.0),
                image: const DecorationImage(
                    fit: BoxFit.cover,

```

```

        image: AssetImage('assets/gif/customer-care.gif'),
    )),
    alignment: Alignment.topLeft,
    margin: const EdgeInsets.all(10),
    padding: const EdgeInsets.all(10),
    child: const Text('Settings',
        style: TextStyle(
            fontFamily: 'OdorMeanChey',
            fontWeight: FontWeight.bold))),
),
),
const SizedBox(height: 10)
]));
}
}

```

main_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/splash_screen_controll
er.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class MainScreen extends StatelessWidget {
    static const String id = "";
    final splashScreenController = Get.put(SplashScreenController());
    MainScreen({super.key});

    @override
    Widget build(BuildContext context) {

```

```

return GetX<SplashScreenController>(
    init: splashScreenController,
    builder: (controller) {
        return Scaffold(
            body: Container(
                alignment: Alignment.center,
                decoration: const BoxDecoration(color: Colors.white),
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                        Image.asset('assets/images/people.jpg'),
                        Container(
                            margin: const EdgeInsets.all(45),
                            child: controller.isLoading.value
                                ? const Text(
                                    'APLIKASI KESEHATAN MENTAL ANAK REMAJA',
                                    textAlign: TextAlign.center,
                                    style: TextStyle(
                                        fontFamily: 'Barriecito',
                                        color: Color.fromRGBO(255, 255, 255, 1),
                                        fontSize: 48))
                                : Container(
                                    margin: const EdgeInsets.only(top: 50),
                                    child: ElevatedButton(
                                        style: ElevatedButton.styleFrom(
                                            backgroundColor: const Color.fromRGBO(
                                                255, 253, 208, 1),
                                            shape: RoundedRectangleBorder(
                                                borderRadius:
                                                    BorderRadius.circular(50)),
                                        ),
                                        onPressed: () async =>

```

```

        await splashScreenController
            .setIsViewed(1),
        child: const Padding(
            padding: EdgeInsets.only(
                left: 50,
                right: 50,
                top: 10,
                bottom: 10),
        child: Text(
            'GET STARTED',
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                color:
                    Color.fromRGBO(0, 0, 0, 1),
                fontWeight: FontWeight.w800,
            ),
        ))),
    ))));
},
);
}
}

```

music_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/audio_controller.d
art';

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';


```

```

class MusicScreen extends StatelessWidget {
    const MusicScreen(this.music, {super.key});

```

```

final Music? music;

@Override
Widget build(BuildContext context) {
    return GetBuilder<MusicController>(builder: (controller) {
        return WillPopScope(
            onWillPop: () async {
                controller.disposeAudioPlayer();
                return true;
            },
            child: Scaffold(
                body: Padding(
                    padding: const EdgeInsets.all(20),
                    child: Column(
                        mainAxisAlignment: MainAxisAlignment.center,
                        children: [
                            music!.musicCover != null || music!.musicCover != ""
                                ? ClipRRect(
                                    borderRadius: BorderRadius.circular(20),
                                    child: Image.network(
                                        music!.musicCover.toString(),
                                        width: double.infinity,
                                        height: 350,
                                        fit: BoxFit.cover,
                                        errorBuilder: (BuildContext context,
                                            Object exception,
                                            StackTrace? stackTrace) {
                                            return Center(
                                                child: Image.asset(
                                                    "assets/images/default-cover-music.jpg"),
                                            );
                                        },
                                    ),
                                )
                            : Center(
                                child: Image.asset(
                                    "assets/images/default-cover-music.jpg"),
                            ),
                        ],
                    ),
                ),
            ),
        );
    });
}

```

```

),
)
: ClipRRect(
    borderRadius: BorderRadius.circular(20),
    child: Text(music!.musicName.toString()),
),
const SizedBox(height: 10),
Text(music!.musicName.toString()),
const SizedBox(height: 10),
Padding(
    padding: const EdgeInsets.symmetric(horizontal: 16),
    child: Row(
        mainAxisAlignment:
            MainAxisAlignment.spaceBetween,
        children: [
            Text(controller.formatTime(
                controller.position.value.inSeconds)),
            Text(controller.formatTime(
                (controller.duration.value -
                    controller.position.value)
                .inSeconds))
        ],
    )));
Slider(
    value: controller.position.value.inSeconds.toDouble(),
    min: 0.0,
    max: controller.duration.value.inSeconds.toDouble(),
    onChanged: (value) async {
        controller.slider(value);
    },
),
CircleAvatar(
    radius: 35,
    child: IconButton(

```

```
        icon: Icon(controller.isPlaying.value
            ? Icons.pause
            : Icons.play_arrow),
        onPressed: () async {
            await controller.onClickEvent();
        },
    ),
)
]
)));
}
}
```

profile_screen.dart

```
import 'dart:io';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/profile_controller.
dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller
.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class ProfileScreen extends StatelessWidget {
ProfileScreen({super.key});
static const String id = "profile_screen";

final userGuardsController = Get.put(UserGuardsController());
```

```

final profileController = Get.put(ProfileController());

@Override
Widget build(BuildContext context) {
    return GetBuilder<ProfileController>(
        init: profileController,
        builder: (controller) {
            return Scaffold(
                body: Container(
                    decoration: const BoxDecoration(
                        gradient: LinearGradient(
                            begin: Alignment.topRight,
                            end: Alignment.bottomLeft,
                            colors: [
                                Color.fromRGBO(255, 253, 208, 1),
                                Color.fromRGBO(255, 255, 255, 1),
                            ])),
                child: ListView(
                    padding: EdgeInsets.zero,
                    children: <Widget>[
                        Padding(
                            padding: const EdgeInsets.all(5),
                            child: Column(children: [
                                Column(
                                    mainAxisAlignment: MainAxisAlignment.center,
                                    children: [
                                        const SizedBox(height: 50),
                                        controller.imageFile.value != null ||
                                            controller.usernameController.value.text !=
                                                ""
                                            ? CircleAvatar(
                                                radius: 100,
                                                backgroundColor: Colors.brown,
                                            )
                                        : Container()
                                    ],
                                ),
                            ],
                        ),
                    ],
                ),
            );
        }
    );
}

```

```

backgroundImage: FileImage(File(controller
    .imageFile.value!.path
    .toString())),
child: Text(controller.usernameController.value.text,
    style: const TextStyle(
        color: Colors.white)))
: CircleAvatar(
    radius: 100,
    backgroundColor: Colors.brown,
    backgroundImage: NetworkImage(
        userGuardsController
            .user.currentUser!.photoURL
            .toString()),
    child: Text(
        userGuardsController.user.currentUser!.displayName.toString(),
        style: const TextStyle(color: Colors.white))),
Text(
    userGuardsController
        .user.currentUser!.displayName!
        .toString(),
    style: const TextStyle(
        fontFamily: 'OdorMeanChey',
        fontWeight: FontWeight.bold,
        color: Colors.black)),
Text(
    userGuardsController.user.currentUser!.email!
    .toString(),
    style: const TextStyle(
        fontFamily: 'OdorMeanChey',
        fontWeight: FontWeight.bold,
        color: Colors.black)),
const SizedBox(height: 50),

```

```

const Divider(),
Form(
  key: controller.formKeyUpdateProfile.value,
  child: Column(children: [
    Padding(
      padding: const EdgeInsets.all(5),
      child: SizedBox(
        width: double.infinity,
        child: ElevatedButton(
          style: ElevatedButton.styleFrom(
            fixedSize:
              const Size(100, 50),
            shape: RoundedRectangleBorder(
              borderRadius:
                BorderRadius.circular(
                  30),
            ),
            backgroundColor:
              const Color.fromRGBO(
                255, 220, 220, 1)),
          onPressed: controller.pickImage,
          child: const Text('Pick Image',
            style: TextStyle(
              fontFamily:
                'OdorMeanChey',
              fontWeight:
                FontWeight.bold,
              color: Colors.black)),
        )));
    Padding(
      padding: const EdgeInsets.all(5),
      child: TextFormField(
        validator: (value) {

```



```

        }

        return null;
    },
    decoration: InputDecoration(
        labelText: 'Username',
        labelStyle: const TextStyle(
            fontFamily: 'OdorMeanChey',
            fontWeight: FontWeight.bold,
            color: Colors.black),
        filled: true,
        fillColor: const Color.fromRGBO(
            255, 220, 220, 1),
        border: OutlineInputBorder(
            borderRadius:
                BorderRadius.circular(30),
        ),
        ),
        )),
    Padding(
        padding: const EdgeInsets.all(5),
        child: SizedBox(
            width: double.infinity,
            child: ElevatedButton(
                style: ElevatedButton.styleFrom(
                    fixedSize:
                        const Size(100, 50),
                    backgroundColor:
                        const Color.fromRGBO(
                            255, 220, 220, 1)),
                onPressed: () async {
                    if (controller
                        .formKeyUpdateProfile
                        .value

```

```

        .currentState!
        .validate() {
            await controller
                .updateUserProfile();
        }
    },
    child: const Text(
        'Update Profile',
        style: TextStyle(
            fontFamily:
                'OdorMeanChey',
            fontWeight:
                FontWeight.bold,
            color: Colors.black)),
    )),
    const Divider(),
),
],
),
),
),
);
);
}
}

```

search_music_screen.dart

```

import 'package:aplikasi_kesehatan_mental_anak_remaja/models/music.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import '../get_x/controllers/audio_controller.dart';

```

```

class SearchMusicScreen extends StatelessWidget {
  const SearchMusicScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<MusicController>(builder: (controller) {
      return WillPopScope(
        onWillPop: () async {
          controller.clearSearch();
          return true;
        },
        child: Scaffold(
          appBar: AppBar(
            title: const Text('See All Music',
              style: TextStyle(
                fontFamily: 'OdorMeanChey',
                fontWeight: FontWeight.bold,
                color: Color.fromRGBO(221, 56, 56, 1))),
            automaticallyImplyLeading: false,
            centerTitle: true,
            backgroundColor: const Color.fromRGBO(255, 220, 220, 1),
          ),
          backgroundColor: const Color.fromRGBO(255, 253, 208, 1),
          body: Column(
            children: [
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: TextField(
                  onChanged: (value) {
                    controller.setSearch(value);
                  },
                  decoration: const InputDecoration(

```

```

        labelText: 'Search',
        hintText: 'Search for music',
        prefixIcon: Icon(Icons.search),
      ),
    ),
  ),
Expanded(
  child: StreamBuilder<List<Music>>(
    stream: controller.searchMusic(),
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Center(
          child: CircularProgressIndicator(),
        );
      }

      if (snapshot.hasError) {
        return Center(
          child: Text('Error: ${snapshot.error}'),
        );
      }

      if (!snapshot.hasData || snapshot.data!.isEmpty) {
        return const Center(
          child: Text('No Video found.'),
        );
      }

      return GridView.builder(
        gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
          crossAxisCount: 2,
          crossAxisSpacing: 8.0,
          mainAxisSpacing: 8.0,
        )
      );
    }
  );
}

```

```
),
itemCount: snapshot.data!.length,
itemBuilder: (context, index) {
  List<Music>? musics = snapshot.data;
  return Padding(
    padding: const EdgeInsets.all(10),
    child: GestureDetector(
      onTap: () => controller.setAudio(musics[index]),
      child: Container(
        width: 150,
        decoration: BoxDecoration(
          borderRadius: BorderRadius.circular(8.0),
          image: DecorationImage(
            fit: BoxFit.cover,
            image: NetworkImage(
              musics![index].musicCover.toString(),
            ),
          ),
        ),
        onError: (exception, stackTrace) {
          const Center(
            child: Text("Image Error"),
          );
        },
      ),
    ),
    padding: const EdgeInsets.all(5),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.end,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        SizedBox(
          width: 150,
          child: Text(
            musics[index].musicName.toString(),
          ),
        ),
      ],
    ),
  );
}
```

```

        style: const TextStyle(
            color: Colors.black,
            fontSize: 16,
            fontWeight: FontWeight.bold,
        ),
        maxLines: 2,
        overflow: TextOverflow.ellipsis,
    ),
),
],
),
),
),
),
),
);
},
);
),
),
),
],
),
));
},
);
}
}

```

search_video_screen.dart

```

import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/video_controller.d
art';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

```

```
import '../../models/video.dart';

class SearchVideoScreen extends StatelessWidget {
  const SearchVideoScreen({super.key});

  @override
  Widget build(BuildContext context) {
    return GetBuilder<VideoController>(builder: (controller) {
      return WillPopScope(
        onWillPop: () async {
          controller.clearSearch();
          return true;
        },
        child: Scaffold(
          appBar: AppBar(
            title: const Text('See All Video',
              style: TextStyle(
                fontFamily: 'OdorMeanChey',
                fontWeight: FontWeight.bold,
                color: Color.fromRGBO(221, 56, 56, 1))),
            automaticallyImplyLeading: false,
            centerTitle: true,
            backgroundColor: const Color.fromRGBO(255, 220, 220, 1),
          ),
          backgroundColor: const Color.fromRGBO(255, 253, 208, 1),
          body: Column(
            children: [
              Padding(
                padding: const EdgeInsets.all(8.0),
                child: TextField(
                  onChanged: (value) {
                    controller.setSearch(value);
                  },
                ),
              ),
            ],
          ),
        ),
      );
    });
  }
}
```

```
decoration: const InputDecoration(  
    labelText: 'Search',  
    hintText: 'Search for video',  
    prefixIcon: Icon(Icons.search),  
,  
,  
,  
    Expanded(  
        child: StreamBuilder<List<Video>>(  
            stream: controller.searchVideo(),  
            builder: (context, snapshot) {  
                if (snapshot.connectionState == ConnectionState.waiting) {  
                    return const Center(  
                        child: CircularProgressIndicator(),  
                    );  
                }  
  
                if (snapshot.hasError) {  
                    return Center(  
                        child: Text('Error: ${snapshot.error}'),  
                    );  
                }  
  
                if (!snapshot.hasData || snapshot.data!.isEmpty) {  
                    return const Center(  
                        child: Text('No Video found.'),  
                    );  
                }  
  
                return GridView.builder(  
                    gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(  
                        crossAxisCount: 2,  
                        crossAxisSpacing: 8.0,
```

```

        mainAxisSpacing: 8.0,
      ),
      itemCount: snapshot.data!.length,
      itemBuilder: (context, index) {
        List<Video>? videos = snapshot.data;
        return Padding(
          padding: const EdgeInsets.all(10),
          child: GestureDetector(
            onTap: () => controller.setVideo(videos[index]),
            child: Container(
              width: 150,
              decoration: BoxDecoration(
                borderRadius: BorderRadius.circular(8.0),
                image: DecorationImage(
                  fit: BoxFit.cover,
                  image: NetworkImage(
                    videos![index]
                    .videoCaptionUrl
                    .toString(),
                  ),
                ),
                onError: (exception, stackTrace) {
                  const Center(
                    child: Text("Image Error"),
                  );
                },
              ),
            ),
            padding: const EdgeInsets.all(5),
            child: Column(
              mainAxisAlignment: MainAxisAlignment.end,
              crossAxisAlignment: CrossAxisAlignment.start,
              children: [
                SizedBox(

```

```
        width: 150,  
        child: Text(  
            videos[index].videoTitle.toString(),  
            style: const TextStyle(  
                color: Colors.black,  
                fontSize: 16,  
                fontWeight: FontWeight.bold,  
            ),  
            maxLines: 2,  
            overflow: TextOverflow.ellipsis,  
        ),  
        ),  
    ],  
),  
),  
),  
),  
);  
},  
);  
},  
),  
),  
],  
),  
));  
});  
}  
}
```

setting_screen.dart

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/auth_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/guards/user_guards_controller.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/crisis_support_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/diagnose_history_screen.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/user_screen/profile_screen.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';

class SettingScreen extends StatelessWidget {
    static const String id = "setting_screen";

    final authController = Get.put(AuthController());
    final userGuardsController = Get.put(UserGuardsController());

    SettingScreen({super.key});

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: Container(
                decoration: const BoxDecoration(
                    gradient: LinearGradient(

```

```
begin: Alignment.topRight,  
end: Alignment.bottomLeft,  
colors: [  
    Color.fromRGBO(255, 253, 208, 1),  
    Color.fromRGBO(255, 255, 255, 1),  
]),  
child: ListView(padding: const EdgeInsets.all(10), children: <Widget>[  
    Padding(  
        padding: const EdgeInsets.all(5),  
        child: Column(children: [  
            const SizedBox(height: 100),  
            CircleAvatar(  
                radius: 100,  
                backgroundColor: Colors.brown,  
                backgroundImage: NetworkImage(userGuardsController  
                    .user.currentUser!.photoURL  
                    .toString()),  
            ),  
            child: Text(  
                userGuardsController.user.currentUser!.displayName  
                .toString(),  
                style: const TextStyle(color: Colors.white))),  
            const SizedBox(height: 100),  
            Container(  
                padding: const EdgeInsets.all(5),  
                margin: const EdgeInsets.all(5),  
                child: ElevatedButton(  
                    style: ElevatedButton.styleFrom(  
                        backgroundColor: const Color.fromRGBO(255, 220, 220, 1),  
                        alignment: Alignment.center,  
                        shape: RoundedRectangleBorder(  
                            borderRadius: BorderRadius.circular(20),  
                        )),  
                ),  
            ),  
        ],  
    ),  
],
```

```

 onPressed: () => Get.to(ProfileScreen()),
 child: const Row(
 mainAxisAlignment: MainAxisAlignment.spaceBetween,
 children: [
 Padding(
 padding: EdgeInsets.all(10),
 child: Text('Profile',
 style: TextStyle(
 fontFamily: 'MochiyPopOne',
 fontSize: 20,
 color: Colors.black)),
 ),
 ],
 )),
),
Container(
 padding: const EdgeInsets.all(5),
 margin: const EdgeInsets.all(5),
 child: ElevatedButton(
 style: ElevatedButton.styleFrom(
 backgroundColor: const Color.fromRGBO(255, 220, 220, 1),
 alignment: Alignment.center,
 shape: RoundedRectangleBorder(
 borderRadius: BorderRadius.circular(20)),
 ),
 ),
),
),
onPressed: () => Get.to(DiagnoseHistoryScreen()),
child: const Row(
 mainAxisAlignment: MainAxisAlignment.spaceBetween,
 children: [
 Padding(
 padding: EdgeInsets.all(10),
 child: Text('Diagnose History',

```

```
        style: TextStyle(  
            fontFamily: 'MochiyPopOne',  
            fontSize: 20,  
            color: Colors.black)),  
    ),  
],  
)),  
,  
Container(  
    padding: const EdgeInsets.all(5),  
    margin: const EdgeInsets.all(5),  
    child: ElevatedButton(  
        style: ElevatedButton.styleFrom(  
            backgroundColor: const Color.fromRGBO(255, 220, 220, 1),  
            alignment: Alignment.center,  
            shape: RoundedRectangleBorder(  
                borderRadius: BorderRadius.circular(20),  
            ),  
        ),  
        onPressed: () => Get.to(CrisisSupportScreen()),  
        child: const Row(  
            mainAxisAlignment: MainAxisAlignment.spaceBetween,  
            children: [  
                Padding(  
                    padding: EdgeInsets.all(10),  
                    child: Text('Consultation',  
                        style: TextStyle(  
                            fontFamily: 'MochiyPopOne',  
                            fontSize: 20,  
                            color: Colors.black))),  
            ],  
        )),  
,
```

```
Container(  
    padding: const EdgeInsets.all(5),  
    margin: const EdgeInsets.all(5),  
    child: ElevatedButton(  
        style: ElevatedButton.styleFrom(  
            backgroundColor: const Color.fromRGBO(255, 220, 220, 1),  
            alignment: Alignment.center,  
            shape: RoundedRectangleBorder(  
                borderRadius: BorderRadius.circular(20),  
            ),  
        ),  
        onPressed: () async => await authController.logout(),  
        child: const Row(  
            mainAxisAlignment: MainAxisAlignment.spaceBetween,  
            children: [  
                Padding(  
                    padding: EdgeInsets.all(10),  
                    child: Text('Sign Out',  
                        style: TextStyle(  
                            fontFamily: 'MochiyPopOne',  
                            fontSize: 20,  
                            color: Colors.black))),  
            ],  
        )),  
    ),  
],  
,  
);  
}  
}
```

```
import
'package:aplikasi_kesehatan_mental_anak_remaja/get_x/controllers/video_controller.d
art';
import 'package:aplikasi_kesehatan_mental_anak_remaja/models/video.dart';
import
'package:aplikasi_kesehatan_mental_anak_remaja/view/widgets/fullscreen_video_widg
et.dart';
import 'package:flutter/material.dart';
import 'package:get/get.dart';
import 'package:video_player/video_player.dart';

class VideoWatchScreen extends StatelessWidget {
    static const String id = "video_watch_screen";
    VideoWatchScreen(this.video, {super.key});

    final videoController = Get.put(VideoController());
    final Video? video;
    @override
    Widget build(BuildContext context) {
        return GetBuilder<VideoController>(
            init: videoController,
            builder: (controller) {
                return WillPopScope(
                    onWillPop: () async {
                        if (controller.videoPlayerController.value.isInitialized) {
                            controller.disposeVideoPlayer();
                            return true;
                        }
                        return false;
                    },
                    child: Scaffold(
                        body: Container(

```

```

decoration: const BoxDecoration(
  gradient: LinearGradient(
    begin: Alignment.topCenter,
    end: Alignment.bottomCenter,
    colors: [
      Color.fromRGBO(255, 253, 208, 1),
      Color.fromRGBO(255, 255, 255, 1),
    ]),
  child: video != null
    ? controller.setFullScreen.value
    ? FullScreenVideo(controller)
    : ListView(children: [
      Container(
        margin:
          const EdgeInsets.only(top: 10, bottom: 10),
        width: double.infinity,
        child: FutureBuilder(
          future:
            controller.initializeVideoPlayerFuture,
          builder: (context, snapshot) {
            if (snapshot.connectionState ==
              ConnectionState.done) {
              return AspectRatio(
                aspectRatio: controller
                  .videoPlayerController
                  .value
                  .aspectRatio,
                child: VideoPlayer(
                  controller.videoPlayerController),
            );
            } else {
              return const Center(
                child: CircularProgressIndicator(),
              );
            }
          },
        ),
      ],
    )));
  }
}

```

```

);
}
}),
),
const SizedBox(width: 20),
Row(
  mainAxisAlignment:
    MainAxisAlignment.spaceBetween,
  children: [
    Padding(
      padding: const EdgeInsets.all(5),
      child: ElevatedButton(
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.red),
        onPressed: controller.onClickEvent,
        child: controller.isVideoPlaying.value
            ? const Icon(Icons.pause)
            : const Icon(Icons.play_arrow))),
    const SizedBox(width: 5),
    Obx(() {
      return Row(children: [
        Text(
          controller.formatTimeVideo(controller
              .position.value.inSeconds),
          style: const TextStyle(fontSize: 5)),
        const VerticalDivider(),
        Text(
          controller.formatTimeVideo(
            (controller.duration.value -
            controller.position.value)
            .inSeconds),
          style: const TextStyle(fontSize: 5))
      ]);
    });
  ],
);

```

```

}),
Obx(() {
return Slider(
    activeColor: Colors.red,
    inactiveColor: const Color.fromARGB(
        199, 123, 121, 120),
    secondaryActiveColor: Colors.red,
    thumbColor: Colors.red,
    value: controller
        .position.value.inSeconds
        .toDouble(),
    min: 0.0,
    max: controller.duration.value.inSeconds
        .toDouble(),
    onChanged: (double newValue) {
        controller.slider(newValue);
    });
}),
IconButton(
    onPressed: () =>
        controller.setFullScreenVideo(),
    icon: const Icon(Icons.fullscreen))
],
),
Container(
    decoration: const BoxDecoration(
        borderRadius: BorderRadius.only(
            topLeft: Radius.circular(20),
            topRight: Radius.circular(20)),
        // color: const Color.fromRGBO(255, 255, 255, 1)
    ),
    child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [

```

```

Padding(
    padding: const EdgeInsets.all(20),
    child: Text(video!.videoTitle ?? 'NA',
        textAlign: TextAlign.left)),

Padding(
    padding: const EdgeInsets.all(20),
    child: Text(
        video!.videoDescription ?? 'NA',
        textAlign: TextAlign.left)),
),

],
),
]

: const Align(
    alignment: Alignment.center,
    child: Padding(
        padding: EdgeInsets.all(20),
        child: Text('404 Not Found',
            style: TextStyle(
                fontFamily: 'MochiyPopOne',
                fontSize: 50)))),
));

});

}
}

```

BAB V

KESIMPULAN DAN SARAN

a. Kesimpulan

Aplikasi Sistem Pakar Diagnosa Kesehatan Mental Remaja Menggunakan Metode Forward Chaining dan Certainty Factor dapat disimpulkan sebagai berikut :

1. Sistem Pakar Diagnosa Kesehatan Mental Remaja Menggunakan Metode Forward Chaining dan Certainty Factor dapat menentukan tingkat depresi berdasarkan gejala-gejala yang dipilih oleh user. Tingkat depresi yang dihasilkan sistem yaitu normal, ringan, sedang dan berat.
2. Sistem Pakar Diagnosa Kesehatan Mental Remaja Menggunakan Metode Forward Chaining dan Certainty Factor memiliki tingkat akurasi yaitu 86,67% dari 15 data yang dimiliki pakar 13 data sama dengan sistem
3. Sistem ini juga memberikan dan mempermudah akses bagi pengguna dalam pengambilan langkah selanjutnya yang perlu dilakukan, yaitu jika pengguna ingin melakukan konsultasi dengan psikolog, maka pengguna dapat memanfaatkan fitur yang telah disediakan.

b. Saran

Saran yang diperoleh diberi untuk penelitian selanjutnya antara lain:

1. Menambahkan kebutuhan fungsional untuk dapat melakukan pemeriksaan secara berkala untuk tahap awal penyembuhan.
2. Menerapkan mekanisme umpan balik yang aktif dan mengumpulkan data evaluasi secara terus-menerus untuk memahami perubahan kebutuhan pengguna dan meningkatkan aplikasi seiring waktu
3. Integrasi dengan layanan kesehatan mental eksternal atau terapis daring untuk memberikan pengguna akses lebih lanjut ke dukungan profesional jika diperlukan.

DAFTAR PUSTAKA

- A.P. Nurabsharina et al., “Aplikasi Sistem Pakar Diagnosis Tingkat Depresi,” vol. 25, no. 1, pp. 76–85, 2020.
- A.Rahmadhani, F. Fauziah, and A. Aningsih, “Sistem Pakar Deteksi Dini Kesehatan Mental Menggunakan Metode Dempster-Shafer,” Sisfotenika, vol. 10, no. 1, p. 37, 2020, doi: 10.30700/jst.v10i1.747.
- L.Sudarmana et al., “Aplikasi Sistem Pakar Untuk mendiagnosis Gangguan Jiwa Schizophrenia,” J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya, vol. 2, no. 2, pp. 40–44, 2018, [Online]. Available: <http://ejournal.poltekegal.ac.id/index.php/informatika/article/download/650/639>
- D.Firmansyah, “Sistem Pakar Diagnosa Awal Gangguan Jiwa Menggunakan Media Mobile Seluler Menggunakan Media Mobile Seluler,” 2011.
- Adisty Wismani Putri, Budhi Wibhawa, Arie Surya Gutama. 2015. Kesehatan Mental Masyarakat Indonesia (Pengetahuan dan Keterbukaan Masyarakat terhadap Gangguan Kesehatan Mental)
- Anindito Aditomo, Sofia Retnowati. 2004. Perfeksionisme, Harga Diri, dan Kecenderungan Depresi Pada Remaja Akhir. Yogyakarta: Fakultas Psikologi Universitas Gajah Mada
- Apip Supiandi, Damar Bagja Chandradimuka. 2017. Sistem Pakar Diagnosa Depresi Mahasiswa Akhir Dengan Metode Certainty Factor Berbasis Mobile. Sukabumi: STMIK Nusa Mandiri Sukabumi



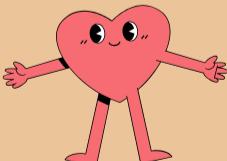
SELF LOVE AND MENTAL HEALTH



Application Purpose

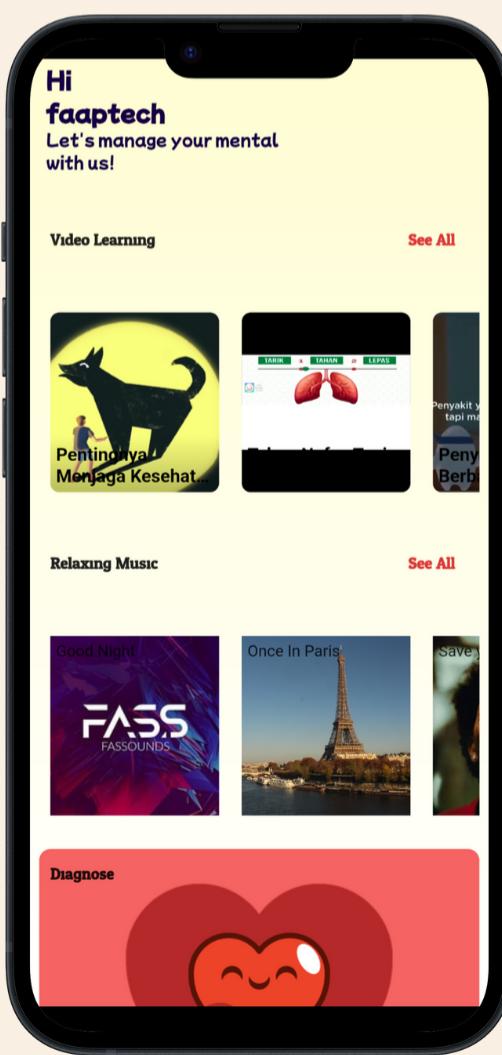
Helping teenagers overcome problems and providing initial treatment assistance regarding mental health in teenagers.

Application Benefits



1. Diagnosing someone's Mental Health.
2. Providing awareness to people about Mental Health.
3. Providing Information and Assistance Services.
4. Make it easier to consult online.

Find Balance, Achieve Your Dreams!



Application Features

1. User Authentication
2. Profile Management
3. Mental Health Assessment
4. Crisis Support
5. Resource Library
6. Journalism Features
7. Appointment Scheduling
8. Push Notifications