PENGEMBANGAN *BACK-END* APLIKASI MENTALWELL 1.0 BERBASIS *WEBSITE* MENGGUNAKAN *FRAMEWORK* HAPI.JS

(Skripsi)

Oleh

ANINDYA KINARYA YANG ESA RIYANTO

NPM 2115061013

FAKULTAS TEKNIK

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2025

PENGEMBANGAN *BACK-END* APLIKASI MENTALWELL 1.0 BERBASIS *WEBSITE* MENGGUNAKAN *FRAMEWORK* HAPI.JS

Oleh
ANINDYA KINARYA YANG ESA RIYANTO

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar
SARJANA TEKNIK

Pada

Program Studi Teknik Informatika
Jurusan Teknik Elektro
Fakultas Teknik Universitas Lampung

FAKULTAS TEKNIK
UNIVERSITAS LAMPUNG
BANDAR LAMPUNG
2025

ABSTRAK

PENGEMBANGAN *BACK-END* APLIKASI MENTALWELL 1.0 BERBASIS *WEBSITE* MENGGUNAKAN *FRAMEWORK* HAPI.JS

Oleh

Anindya Kinarya Yang Esa Riyanto

Pengembangan aplikasi MentalWell 1.0 dilakukan untuk menghadirkan layanan konseling psikologis berbasis web yang dapat diakses masyarakat. Fokus penelitian ini adalah pada pengembangan sistem *back-end* menggunakan *framework* Hapi.js untuk mengimplementasikan API berarsitektur REST dengan struktur modular dan pengelolaan *routing* yang terorganisir, sehingga memudahkan proses pengembangan maupun pemeliharaan kode. Keamanan sistem menerapkan JSON Web Token (JWT) sebagai mekanisme autentikasi dan otorisasi berbasis peran. Proses pengembangan dilakukan dengan metode *Agile* melalui enam iterasi, yang terdiri dari pembangunan fitur dasar untuk pasien dan psikolog, penambahan fungsionalitas administrator, pengembangan sistem pemesanan konseling, penerapan mekanisme *soft delete*, serta perbaikan dan optimalisasi fitur berdasarkan umpan balik hasil integrasi dengan front-end. Hasil akhir pengembangan menghasilkan 56 modul dengan 36 *endpoint* untuk peran publik, pasien, psikolog, dan administrator. Seluruh *endpoint* diuji menggunakan Postman dengan total 122 *test case* yang menunjukkan tingkat keberhasilan 100%, dan sistem *back-end* telah terintegrasi dengan *front-end* untuk mendukung layanan konseling daring secara penuh.

Kata kunci: Kesehatan Mental, Konseling Daring, REST API, Hapi.js, JSON Web Token, *Agile*

ABSTRACT

BACK-END DEVELOPMENT OF THE WEB-BASED APPLICATION MENTALWELL 1.0 USING THE HAPI.JS FRAMEWORK

By

Anindya Kinarya Yang Esa Riyanto

The development of MentalWell 1.0 was carried out to provide accessible web-based psychological counseling services for the public. This research focuses on the back-end system development using the Hapi.js framework to implement a REST-architectured API with a modular structure and organized routing management, thereby facilitating both development and maintenance processes. Security is ensured through the application of JSON Web Token (JWT) as the mechanism for role-based authentication and authorization.. The development process was conducted using the Agile method through six iterations, consisting of building basic features for patients and psychologists, adding administrator functionalities, developing the counseling booking system, implementing the soft delete mechanism, as well as improving and optimizing features based on feedback from front-end integration. The final outcome produced 56 modules with 36 endpoints for public, patient, psychologist, and administrator roles. All endpoints were tested using Postman with a total of 122 test cases that achieved a 100% success rate, and the back-end system has been fully integrated with the front-end to support online counseling services.

Keywords: Mental Health, Online Counseling, REST API, Hapi.js, JSON Web Token, Agile

Judul Skripsi	:	PENGEMBANGAN *BACK-END* APLIKASI MENTALWELL 1.0 BERBASIS *WEBSITE* MENGGUNAKAN *FRAMEWORK* HAPI.JS
Nama Mahasiswa	:	Anindya Kinarya Yang Esa Riyanto
Nomor Pokok Mahasiswa	:	2115061013
Program Studi	:	Teknik Informatika
Jurusan	:	Teknik Elektro
Fakultas	:	Teknik

MENYETUJUI

1. Komisi Pembimbing

Pembimbing Utama	Pembimbing Pendamping
Ir. Trisya Septiana, S.T., M.T., IPM	Ir. Resty Annisa, S.ST., M.Kom
NIP. 199009212019032025	NIP. 199008302019032019

2. Mengetahui

Ketua Jurusan Teknik Elektro	Ketua Program Studi Teknik Informatika
Herlinawati, S.T., M.T.	Yessi Mulyani, S.T., M.T.
NIP. 197103141999032001	NIP. 197312262000122001

MENGESAHKAN

1. Tim Penguji

Ketua : **Ir. Trisya Septiana, S.T., M.T., IPM** .....

Sekretaris : **Ir. Resty Annisa, S.ST., M.Kom** .....

Penguji : **Ir. Gigih Forda Nama, S.T., M.T.I., IPM** .....

2. Dekan Fakultas Teknik

Dr. Eng. Ir. Helmy Fitriawan, S.T., M.Sc.
NIP 197509282001121002

Tanggal Lulus Ujian Skripsi: **21 Agustus 2025**

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi berjudul "Pengembangan *Back-End* Aplikasi MentalWell 1.0 Berbasis *Website* Menggunakan *Framework* Hapi.js" sepenuhnya merupakan hasil karya saya sendiri. Apabila di kemudian hari terbukti pernyataan ini tidak benar, saya siap menerima sanksi sesuai dengan ketentuan hukum yang berlaku.

Bandar Lampung, 21 Agustus 2025

Penulis,

Anindya Kinarya Yang Esa Riyanto

NPM. 2115061013

RIWAYAT HIDUP

Penulis lahir di Kota Metro pada 8 April 2003 dan merupakan anak kedua dari pasangan Bapak Suis Riyanto dan Ibu Ida Rismawati. Penulis memulai pendidikannya di TK Tunas Bangsa Bratasena Adiwarna, kemudian melanjutkan ke SDIT Insan Cendekia Pasiran Jaya dan lulus pada tahun 2015. Pendidikan menengahnya ditempuh di SMP Al-Kautsar Bandar Lampung lulus pada tahun 2018 dan dilanjutkan di SMA Al-Kautsar Bandar Lampung hingga lulus pada tahun 2021. Pada tahun yang sama, penulis melanjutkan studi di Universitas Lampung pada Program Studi Teknik Informatika, Jurusan Teknik Elektro, Fakultas Teknik.

Selain aktif menjadi mahasiswa Teknik Informatika, penulis juga mengikuti berbagai kegiatan pengembangan diri dan profesional, di antaranya mengikuti program Studi Independen Bersertifikat di Bangkit Academy pada tahun 2023 untuk mendalami bidang teknologi, mendapatkan pengalaman profesional melalui program Merdeka Belajar Kampus Merdeka (MBKM) di PT United Tractors Tbk dengan posisi sebagai IT & *Data Science Developer*, serta melaksanakan Kuliah Kerja Nyata (KKN) pada tahun 2024 sebagai bentuk pengabdian kepada masyarakat di Desa Umpu Bhakti, Kecamatan Blambangan Umpu, Kabupaten Way Kanan.

MOTTO

"Maka, sesungguhnya beserta kesulitan ada kemudahan. Sesungguhnya beserta kesulitan ada kemudahan."
(Al-Insyirah : 5-6)

"I'm a pendulum, I don't know where I'll land. I just know that I can swing it, I'll always swing it."
(Nicole Zefanya)

"Next time is next time. Now is now."
(*Perfect Days*, dir. Wim Wenders, 2023)

PERSEMBAHAN

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Dengan mengucap syukur kehadirat Allah SWT,
karya sederhana ini kupersembahkan kepada:

"Kedua orang tuaku tercinta, yang selalu menjadi sumber kekuatan dan inspirasiku. Terima kasih atas segala pengorbanan, cinta tanpa syarat, dan doa yang tak pernah putus. Setiap langkah yang aku tempuh adalah buah dari perjuangan kalian, dan semoga ilmu yang aku raih ini menjadi pahala yang terus mengalir untuk kalian."

"Mamasku, Mahatma Ridho Riyanto, dan adikku, Quada Karena Nya Riyanto, yang kehadirannya membawa tawa dan warna dalam hidupku. Semoga kita senantiasa menjaga kebersamaan ini dan membahagiakan kedua orang tua kita di masa depan."

"Keluarga besar Teknik Elektro 2021 dan teman-teman seperjuangan, yang telah berbagi begitu banyak cerita."

"Almamater tercinta, Universitas Lampung dan Jurusan Teknik Elektro"

SANWACANA

Segala puji dan syukur penulis panjatkan kehadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi ini. Shalawat beriring salam semoga senantiasa tercurah kepada teladan kita, Nabi Muhammad SAW, beserta keluarga dan para sahabatnya.

Penyusunan skripsi berjudul "Pengembangan *Back-End* Aplikasi MentalWell 1.0 Berbasis Website Menggunakan *Framework* Hapi.js" ini merupakan tahap akhir studi penulis untuk memenuhi syarat kelulusan dan memperoleh gelar Sarjana Teknik pada Jurusan Teknik Elektro, Fakultas Teknik, Universitas Lampung. Proses penyusunan ini tidak terlepas dari bimbingan dan dukungan berbagai pihak. Oleh karena itu, dengan penuh rasa hormat, penulis ingin menghaturkan terima kasih kepada:

1. Bapak Dr. Eng. Helmy Fitriawan, S.T., M.Sc., selaku Dekan Fakultas Teknik, Universitas Lampung.
2. Kedua orang tua tercinta, Ayahanda Suis Riyanto dan Ibunda Ida Rismawati, terima kasih atas limpahan doa, kasih sayang, dan dukungan yang tak pernah putus.
3. Ibu Herlinawati, S.T., M.T., selaku Ketua Jurusan Teknik Elektro.
4. Ibu Yessi Mulyani, S.T., M.T., selaku Ketua Program Studi Teknik Informatika.
5. Ibu Ir. Resty Annisa, S.ST., M.Kom., selaku Pembimbing Akademik dan Dosen Pembimbing II, atas kesabaran, ilmu, dan waktu yang telah diluangkan.
6. Ibu Ir. Trisya Septiana, S.T., M.T., I.P.M., selaku Dosen Pembimbing I, atas bimbingan, arahan, dan motivasinya selama penyusunan skripsi.
7. Bapak Ir. Gigih Forda Nama, S.T., M.T.I., I.P.M., selaku Dosen Penguji, atas saran dan masukan yang membangun saat sidang skripsi.
8. Seluruh Dosen dan Staf Administrasi Jurusan Teknik Elektro atas ilmu dan bantuan yang diberikan selama masa perkuliahan.
9. Mbak Rika, selaku admin Program Studi Teknik Informatika, yang telah banyak membantu dalam urusan administrasi.

10. Mamas dan adik tersayang, Mahatma Ridho Riyanto dan Quada Karena Nya Riyanto, terima kasih atas doa, semangat, dan keceriaan yang selalu diberikan, serta untuk Imot Xi Fa Cai, si penghilang stres terbaik di kala pusingnya skripsi.
11. Echa Andrea, rekan tim dalam proyek ini, terima kasih atas kepercayaan, kerja sama, dan semua suka duka yang membuat proses ini lebih ringan.
12. Sahabat-sahabat seperjuangan yang telah menjadi bagian penting perjalanan ini:
    - Alya Zahradita Sironi, sahabat terdekat yang selalu menemani dan mendukung sejak awal perkuliahan.
    - Ghefira Shalsabila Calistra, terima kasih telah menjadi teman berangkat ke kampus dan teman berbagi tawa.
    - Agustin Rahmawati, Asima Beatricia, Cella Febriyani, Desti Dian Novera, Saphira Azzahra, dan Siti Nur Azizah atas pertemanan yang luar biasa.
13. Teman-teman Mentee Mascip: Amalia, Annisa, Chelly, Gio, Tiansi, Syifa, dan Veitra, terima kasih atas kebersamaan dan kekompakan yang seru.
14. Semua pihak lain yang tidak dapat disebutkan satu per satu, terima kasih atas segala dukungannya.

Akhir kata, penulis menyadari skripsi ini masih jauh dari sempurna, sehingga kritik dan saran yang membangun sangat diharapkan. Semoga karya ini dapat bermanfaat bagi pembaca dan pengembangan ilmu pengetahuan. Aamiin Ya Rabbal ‘Alamin.

Bandar Lampung, 21 Agustus 2025
Penulis,

Anindya Kinarya Yang Esa Riyanto

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kesehatan mental memainkan peran penting dalam menciptakan kesejahteraan individu dan masyarakat secara keseluruhan. Kesehatan mental yang baik memungkinkan seseorang untuk berfungsi secara produktif dalam kehidupan sehari-hari, membangun hubungan yang sehat, serta mengatasi tantangan yang datang dalam hidup. Namun, meskipun kesadaran tentang pentingnya kesehatan mental semakin meningkat, masih terdapat berbagai hambatan yang menghalangi individu untuk mendapatkan layanan psikologis yang mereka butuhkan.

Salah satu hambatan utama adalah keterbatasan infrastruktur kesehatan mental, terutama di daerah terpencil atau kurang berkembang. Keterbatasan jumlah tenaga profesional dan fasilitas kesehatan mental membuat individu yang tinggal jauh dari pusat kota kesulitan mengakses layanan psikologis yang memadai. Mereka sering kali harus melakukan perjalanan jauh atau mengeluarkan biaya tambahan untuk mendapatkan dukungan yang dibutuhkan, yang pada akhirnya menjadi penghalang besar bagi mereka untuk memprioritaskan kesehatan mentalnya [1].

Di samping itu, stigma sosial masih menjadi penghalang kuat yang membuat individu enggan mencari bantuan profesional. Banyak orang merasa malu atau khawatir dianggap tidak mampu mengatasi masalah sendiri saat mengakses layanan kesehatan mental, sehingga memilih untuk memendam masalahnya daripada menghadapi kemungkinan penilaian negatif dari lingkungan sekitar. Saat ini, teknologi mempermudah akses layanan kesehatan mental melalui *platform* web dan aplikasi digital. Layanan *telemedicine* memungkinkan pengguna mendiskusikan masalah mental dari rumah, mengurangi stigma sosial, memperluas akses di daerah terpencil [2].

Sejalan dengan manfaat ini, MentalWell hadir sebagai aplikasi web yang menghubungkan pengguna dengan psikolog profesional secara daring, menjadi solusi praktis untuk membantu lebih banyak individu mendapatkan dukungan kesehatan mental tanpa terhalang jarak atau stigma sosial. Namun, aplikasi ini masih memerlukan pengembangan lebih lanjut agar dapat memberikan layanan yang lebih optimal dan sesuai kebutuhan pengguna. Oleh karena itu, penelitian ini bertujuan untuk mengembangkan MentalWell 1.0 dengan menambahkan fitur penting, seperti tes psikologi untuk membantu pengguna mengenali kondisi mental mereka dan *chat* langsung dengan psikolog lewat website agar konsultasi bisa dilakukan dengan lebih mudah.

Pengembangan ini memerlukan perhatian pada aspek teknis, terutama pada pengelolaan data dan konektivitas layanan. *Back-end* berperan penting dalam mengelola informasi pengguna, serta memfasilitasi komunikasi antara pengguna dan psikolog. Pengembangan *back-end* yang terstruktur akan membantu memastikan data diproses dengan baik dan aplikasi dapat berjalan sesuai dengan tujuan.

Framework Hapi.js dipilih sebagai fondasi pengembangan *back-end* MentalWell 1.0 karena kemudahannya dalam mengelola API secara terstruktur. Pendekatan berbasis konfigurasi, Hapi.js lebih rapi, mudah dipelihara, dan fleksibel sesuai kebutuhan aplikasi.

Back-end MentalWell 1.0 juga dirancang untuk mendukung autentikasi pengguna dan penyimpanan data yang terstruktur. Hapi.js dapat diintegrasikan dengan berbagai sistem basis data seperti PostgreSQL untuk memastikan informasi pengguna tersimpan dengan terstruktur. Pemanfaatan *framework* ini, diharapkan aplikasi dapat berjalan stabil dan mampu memenuhi kebutuhan pengguna dalam mengakses layanan konsultasi psikologis secara daring.

1.2 Rumusan Masalah

Rumusan masalah penelitian ini meliputi:

1. Bagaimana penerapan Hapi.js dalam pengembangan *back-end* MentalWell 1.0 untuk mendukung konsultasi psikologis dan manajemen data pengguna?

2. Bagaimana cara menerapkan sistem autentikasi dan otorisasi untuk pengguna aplikasi MentalWell 1.0 berbasis website?
3. Bagaimana perancangan *database* untuk menyimpan dan mengelola data pengguna serta riwayat konsultasi di MentalWell 1.0?
4. Bagaimana cara mengintegrasikan *back-end* dengan *front-end* agar aplikasi MentalWell 1.0 dapat berfungsi sesuai kebutuhan utama dan mendukung interaksi dasar pengguna?

1.3 Tujuan Penelitian

Tujuan utama penelitian ini adalah sebagai berikut:

1. Menerapkan *framework* Hapi.js dalam pengembangan *back-end* MentalWell 1.0 untuk mendukung fitur konsultasi psikologis dan manajemen data pengguna.
2. Menerapkan sistem autentikasi dan otorisasi untuk pengguna aplikasi MentalWell 1.0 berbasis website, guna memastikan hanya pengguna yang berwenang dapat mengakses fitur-fitur tertentu sesuai perannya.
3. Merancang dan mengelola *database* yang terstruktur untuk menyimpan dan mengelola data pengguna serta riwayat konseling pada MentalWell 1.0.
4. Mengembangkan mekanisme integrasi *back-end* dan *front-end* agar aplikasi MentalWell 1.0 dapat berfungsi sesuai kebutuhan utama dan mendukung interaksi dasar pengguna dengan antarmuka aplikasi.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan berbagai manfaat, di antaranya:

1. Bagi Praktisi Psikologi

   Penelitian ini memberikan dukungan kepada psikolog melalui sistem yang mempermudah pengelolaan data pengguna, riwayat konseling, serta layanan berbasis digital yang lebih terorganisir.

2. Bagi Masyarakat

   Penelitian ini memberikan manfaat langsung kepada masyarakat dengan menghadirkan solusi untuk meningkatkan aksesibilitas dan kemudahan dalam mendapatkan layanan psikologis. *Platform* yang dikembangkan diharapkan

dapat membantu individu dalam menangani masalah kesehatan mental dengan lebih efisien.

3. Bagi Penelitian Lebih Lanjut

   Penelitian ini dapat menjadi referensi bagi pengembangan teknologi serupa di masa depan, khususnya dalam membangun sistem aplikasi yang mendukung layanan psikologis berbasis digital.

1.5 Batasan Masalah

Agar penelitian ini lebih terfokus dan dapat diselesaikan dengan baik, beberapa batasan masalah yang diterapkan adalah sebagai berikut:

1. Fokus Pengembangan Back-End

   Penelitian ini hanya berfokus pada pengembangan *back-end* aplikasi MentalWell menggunakan *framework* Hapi.js. Pengembangan *front-end* dan desain antarmuka pengguna tidak menjadi cakupan utama penelitian ini, tetapi hanya akan dibahas dalam konteks integrasi dengan *back-end*.

2. Keamanan Data

   Aspek keamanan dalam penelitian ini mengandalkan fitur dasar bawaan dari Hapi.js, dengan penyesuaian minimal seperti *hashing password* menggunakan bcrypt dan implementasi autentikasi sederhana dengan JWT. Validasi input dilakukan untuk mengurangi risiko *error* dan serangan dasar. Pengamanan lebih lanjut, seperti *rate limiting* dan proteksi CSRF, tidak menjadi fokus penelitian ini.

3. Penggunaan Layanan Gratis

   Penelitian ini hanya menggunakan layanan gratis untuk *hosting, database*, dan *storage*, sehingga terdapat keterbatasan dalam hal kapasitas penyimpanan, kecepatan akses.

4. Pengujian Sistem

   Pengujian yang dilakukan dalam penelitian ini terbatas pada pengujian fungsional *back-end* menggunakan Postman. Pengujian mencakup validasi *endpoint* API untuk memastikan operasi pengelolaan *database* berjalan sesuai kebutuhan. Pengujian terkait performa *back-end* atau evaluasi pengalaman pengguna (UX) tidak menjadi fokus utama penelitian ini.

1.6 Sistematika Penulisan

Penyajian skripsi ini dibagi dalam beberapa bab dengan tujuan untuk mempermudah pencarian informasi yang dibutuhkan, serta menunjukkan penyelesaian penelitian secara sistematis. Pembagian bab tersebut adalah sebagai berikut.

BAB I PENDAHULUAN

Bab ini berisikan Latar Belakang, Rumusan Masalah, Tujuan Penelitian, Manfaat Penelitian, Batasan Masalah, serta Sistematika Penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori yang menjadi dasar pengetahuan yang digunakan dalam penyusunan skripsi, mencakup konsep *back-end*, API, REST, keamanan sistem, layanan psikologi digital, tools yang digunakan, dan penelitian terdahulu yang relevan.

BAB III METODOLOGI PENELITIAN

Bab ini menguraikan mengenai waktu dan tempat penelitian, alat dan bahan penelitian, serta tahapan penelitian yang dilakukan. Tahapan penelitian disusun berdasarkan pendekatan *Agile* yang meliputi perencanaan (*plan*), perancangan (*design*), pengembangan (*development*), pengujian dan dokumentasi (*testing & documentation*), serta *deployment* dan *review*.

BAB IV IMPLEMENTASI DAN PEMBAHASAN

Bab ini berisi tentang implementasi hasil desain dan pembahasan yang dilakukan, mencakup struktur kode, konfigurasi server, struktur *database*, *middleware* keamanan, serta hasil pengembangan pada setiap iterasi.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan dari hasil penelitian yang telah dilakukan serta saran yang dapat diberikan untuk pengembangan lebih lanjut.

DAFTAR PUSTAKA

Bab ini berisi referensi yang digunakan dalam penyusunan skripsi.

LAMPIRAN

Bab ini berisi dokumentasi mengenai proses dan hasil kerja penelitian.

BAB II
TINJAUAN PUSTAKA

2.1 Dasar Teori

2.1.1 *Back-End*

Back-end adalah komponen yang beroperasi "di balik layar" dari sebuah situs web atau aplikasi, memainkan peran penting dalam mengelola logika server, interaksi dengan basis data, dan pengelolaan skrip serta arsitektur sistem [3]. Beberapa bahasa pemrograman yang digunakan dalam pengembangan *back-end* adalah PHP, Ruby, Python, dan JavaScript bersama dengan teknologi Node.js, yang berfokus pada komunikasi antara basis data dan browser.

Kode *back-end* berfungsi sebagai perantara yang mentransfer informasi tersimpan di *database* ke antarmuka pengguna melalui browser, memastikan kelancaran operasi yang tidak langsung terlihat oleh pengguna akhir [3].

2.1.2 *Application Programming Interface* (API)

Application Programming Interface atau API merupakan antarmuka yang disediakan oleh suatu perangkat lunak agar dapat diakses oleh perangkat lunak lain, pengguna, atau dalam konteks API web, oleh sistem lain melalui internet. Perancangan sebuah API sering kali mencerminkan banyak hal tentang program di baliknya, mulai dari model bisnis, fitur yang ditawarkan, hingga kemungkinan adanya *bug*. Meskipun secara teknis API dibuat untuk digunakan oleh program lain, kenyataannya API juga harus mudah dipahami oleh pengembang (manusia) yang akan mengintegrasikannya ke dalam sistem mereka [4].

2.1.3 *Representational State Transfer* (REST)

Representational State Transfer atau REST adalah gaya arsitektur untuk membangun layanan web yang efisien, diperkenalkan oleh Roy Fielding pada

tahun 2000 [5]. REST bukanlah sebuah standar atau protokol melainkan sebuah *architectural style* yang memiliki seperangkat prinsip dan batasan yang harus dipatuhi agar suatu sistem dapat dikatakan RESTful.

REST berfokus pada karateristik utama: (1) *Client-Server Architecture*, memisahkan klien dan server untuk skalabilitas; (2) *Statelessness*, setiap permintaan bersifat mandiri tanpa menyimpan status di server; (3) *Uniform Interface*, menggunakan standar HTTP seperti GET, POST, PUT, dan DELETE untuk mempermudah komunikasi; (4) *Cacheable*, respons dari server dapat di-*cache* untuk meningkatkan efisiensi dan kinerja; dan (5) *Layered System*, memiliki beberapa lapisan yang memungkinkan modularitas dalam pengembangan [5].

REST lebih sering digunakan dalam layanan web yang berfokus pada sumber daya. Artinya, REST menyediakan data atau informasi sebagai layanan utama, bukan kumpulan proses atau aktivitas pengolahan data tersebut. REST cocok digunakan sebagai *back-end* aplikasi web karena cara mengaksesnya mudah dan data yang dikirim dalam format JSON, sehingga ukurannya kecil dan efisien.

REST API menjalankan mekanisme komunikasi seperti halnya aplikasi web, yakni dengan *client* mengirimkan *request* melalui HTTP dan server memberikan *response* kepada *client*.

2.1.4 Keamanan dalam Pengembangan *Back-End*

Keamanan dalam pengembangan *back-end* merupakan aspek yang tidak dapat diabaikan, mengingat banyaknya ancaman dunia maya yang dapat mengeksploitasi kelemahan sistem. Beberapa praktik dan konsep penting terkait keamanan dalam pengembangan *back-end* adalah sebagai berikut [6].

1. Autentikasi dan Otorisasi

   Autentikasi memastikan identitas pengguna yang mengakses sistem, sementara otorisasi menentukan hak akses pengguna tersebut. Implementasi JSON Web Token (JWT) memungkinkan transfer informasi yang aman antar pihak karena data telah ditandatangani secara digital.
2. *Middleware* untuk Keamanan

Middleware membantu mengamankan dan mengoptimalkan proses komunikasi antara klien dan server. Dalam konteks Node.js, *middleware* dapat digunakan untuk verifikasi autentikasi token, pengaturan *header* keamanan, dan log permintaan yang mencurigakan.

3. *Role-Based Access Control* (RBAC)

   RBAC membatasi akses pengguna ke sumber daya berdasarkan peran mereka dalam sistem. Ini membantu dalam pengelolaan izin akses dan mencegah akses tidak sah terhadap data sensitif.

4. Validasi dan Sanitasi

   Data Menghindari input berbahaya merupakan langkah penting dalam mencegah serangan injeksi.

5. Perlindungan terhadap Serangan Umum

   Selain autentikasi dan otorisasi, pengembangan *back-end* harus memperhatikan perlindungan terhadap ancaman umum seperti serangan injeksi SQL, *Cross-Site Scripting* (XSS), dan *Cross-Origin Resource Sharing* (CORS).

2.1.3 Aplikasi Layanan Psikologi

Era digital membawa peningkatan kebutuhan akan pengembangan aplikasi untuk layanan psikologi. Transformasi layanan kesehatan mental melalui teknologi informasi memungkinkan pasien untuk mengakses layanan psikologi secara fleksibel tanpa batasan geografis. Berbagai aspek penting terkait dengan integrasi teknologi dalam layanan kesehatan mental dapat dilihat dari beberapa faktor utama, seperti peningkatan aksesibilitas layanan psikologi, privasi dan keamanan data pengguna, personalisasi layanan digital, serta tantangan teknologi dalam layanan psikologi [7].

1. Peningkatan Aksesibilitas Layanan Psikologi

   Teknologi digital, termasuk *telemental healthcare*, memungkinkan pasien yang berada di daerah terpencil atau memiliki kendala waktu dan tempat untuk tetap mendapatkan layanan konsultasi. Aplikasi berbasis *website* memfasilitasi konsultasi secara virtual sehingga kebutuhan interaksi langsung yang sering kali membatasi akses dapat diminimalkan.

2. Privasi dan Keamanan Data Pengguna

Pengembangan aplikasi konsultasi psikologi menghadapi tantangan utama pada aspek privasi data. Data kesehatan mental memiliki sifat yang sangat sensitif, dan perlindungan data pengguna harus menjadi prioritas utama. Hal ini mencakup enkripsi data serta kepatuhan terhadap regulasi privasi yang relevan.

3. Personalisasi dan Responsivitas Layanan Digital

   Aplikasi kesehatan mental berbasis digital dapat menghadirkan layanan yang dipersonalisasi sesuai kebutuhan pengguna. Hal ini mencakup fitur penjadwalan otomatis, pengelolaan rekam jejak konsultasi, serta pengiriman materi terapi yang dapat diakses kapan saja.

4. Tantangan Teknologi dalam Layanan Psikologi

   Meski teknologi mampu memperluas jangkauan layanan kesehatan mental, ada tantangan terkait kemampuan pengguna dalam mengoperasikan teknologi. Kesenjangan digital ini dapat memperburuk ketidaksetaraan akses layanan kesehatan mental jika tidak dikelola dengan baik.

Pengembangan aplikasi konsultasi psikologi berbasis website merupakan solusi relevan untuk mendukung layanan kesehatan mental di era digital. Keamanan data, personalisasi layanan, dan peningkatan aksesibilitas menjadi faktor penting yang memungkinkan aplikasi tersebut memberikan kontribusi positif bagi upaya menjaga kesehatan mental masyarakat.

2.2 Tools yang Digunakan

2.2.1 *Framework* Hapi.js

Hapi.js adalah *framework open-source* untuk pengembangan aplikasi web dan layanan *back-end* berbasis Node.js, dikembangkan oleh tim web seluler di Walmart Labs dan digunakan oleh perusahaan besar seperti PayPal, Yahoo!, dan Disney [8]. *Framework* ini dikenal dengan pendekatan konfigurasi-sentris yang mengutamakan pengaturan melalui konfigurasi daripada kode imperatif, sehingga meningkatkan keterbacaan dan kemudahan pemeliharaan kode.

Hapi.js memiliki fitur utama seperti modularitas tinggi untuk struktur kode yang lebih terorganisir, dukungan *plugin* yang luas untuk integrasi layanan eksternal, serta manajemen routing yang fleksibel. Dari segi keamanan, Hapi.js

menyediakan dukungan untuk validasi input dan memungkinkan penerapan perlindungan terhadap serangan umum seperti XSS dan CSRF melalui ekosistem plugin yang tersedia [9].

Pengembangan REST API berbasis Node.js sangat dipengaruhi oleh pemilihan framework, karena keputusan tersebut berdampak pada struktur arsitektur, efisiensi proses pengembangan, dan skalabilitas aplikasi. Dua *framework* yang umum digunakan adalah Hapi.js dan Express.js, masing-masing dengan pendekatan dan keunggulan yang berbeda.

Penelitian ini memilih Hapi.js karena *framework* ini mendukung arsitektur modular yang sejalan dengan tujuan pengembangan aplikasi, yakni membangun sistem *back-end* yang terstruktur dan mudah dikembangkan secara bertahap.

Express.js merupakan *framework* yang lebih ringan dan fleksibel dengan pendekatan *middleware-centric*. Popularitasnya yang tinggi didukung oleh ekosistem dan komunitas yang luas, sehingga cocok untuk pengembangan cepat dan proyek yang membutuhkan kebebasan tinggi dalam struktur kode. Meskipun tidak menyediakan banyak fitur secara default, fleksibilitas Express.js memungkinkan *developer* untuk membangun sistem sesuai kebutuhan. Tabel berikut menyajikan perbandingan arsitektural antara keduanya.

Tabel 2.1 Perbandingan Arsitektural Express.js dan Hapi.js [10]

	Express.js	**Hapi.js**
Route Configuration	Pakai *middleware* untuk *caching*, validasi, dll; butuh kode tambahan.	Konfigurasi via objek (mis. *cache*).
Routing	Urutan rute penting, bisa duplikasi, rawan bug.	Deterministik, tanpa duplikasi, urut dari spesifik ke umum.
Built-in Capability	Butuh modul tambahan untuk *caching*, autentikasi, atau validasi. (Lebih fleksibel).	Kaya fitur bawaan, seperti *caching server-side* (mendukung memori, Redis, MongoDB) dan autentikasi. Konfigurasi lebih mudah.

Perbandingan ini menunjukkan bahwa Hapi.js lebih sesuai untuk pengembangan aplikasi dengan arsitektur modular dan kebutuhan fitur *back-end* yang kompleks, sementara Express.js tetap menjadi pilihan solid untuk proyek yang mengutamakan fleksibilitas dan kecepatan pengembangan.

2.2.2 JSON Web Token (JWT)

JSON Web Token atau JWT adalah standar terbuka (RFC 7519) yang digunakan untuk mengirimkan data atau klaim antar pihak dalam format JSON. Implementasi JWT memungkinkan transfer informasi yang aman antar pihak karena data telah ditandatangani secara digital [6].

JWT terdiri dari tiga elemen utama yang dienkripsi secara terpisah, yakni *header*, *payload*, dan *signature*. *Header* mencakup informasi tentang tipe token dan algoritma enkripsi yang digunakan, *payload* berisi data atau klaim yang ingin disampaikan, dan *signature* memanfaatkan kunci rahasia untuk menjaga keamanan serta memverifikasi integritas token [11].

2.2.3 Supabase

Supabase adalah *platform* open-source *back-end-as-a-service* (BaaS) yang menggunakan PostgreSQL sebagai basis data relasional dan menyediakan layanan seperti *PostgreSQL database*, *authentication*, *storage*, *edge function,* dan *realtime*. *Platform* ini menawarkan solusi terintegrasi yang fleksibel dan skalabel sebagai alternatif untuk Firebase. Berikut adalah komponen Supabase yang digunakan dalam penelitian ini.

A. PostgreSQL

PostgreSQL adalah sistem manajemen basis data relasional *open-source* yang mendukung *query* SQL dan transaksi ACID (*Atomicity, Consistency, Isolation, Durability*). ACID memastikan transaksi dijalankan secara atomik, konsisten, terisolasi, dan tahan lama, mendukung pengelolaan data terstruktur dengan performa tinggi [12].

B. *Realtime*

Fitur *Realtime* memungkinkan sinkronisasi data melalui WebSocket dengan memantau perubahan pada tabel *database*. Fitur ini mendukung pembaruan data instan untuk aplikasi yang memerlukan interaksi dinamis [13].

C. *Storage*

Supabase Storage menyediakan sistem penyimpanan objek berbasis *bucket* untuk mengelola *file*, seperti gambar, video, dan dokumen. Berikut komponen dari *Storage*.

1. *Files*

   Berbagai jenis *file* media, termasuk gambar, GIF, dan video. *File* disimpan di luar *database* untuk efisiensi ukuran, dengan *file* HTML dikembalikan sebagai teks biasa untuk keamanan [14].

2. *Folders*

   Struktur untuk mengatur *file* sesuai kebutuhan proyek, serupa dengan folder pada sistem computer [14].

3. *Buckets*

   Wadah utama untuk *file* dan *folder*, memungkinkan pengaturan aturan keamanan dan akses yang berbeda, seperti *bucket* khusus untuk video atau gambar profil [14].

2.2.4 Railway

Railway merupakan *platform cloud* berbasis *Platform as a Service* (PaaS) yang dirancang untuk menyederhanakan proses pengembangan dan *deployment* aplikasi. Railway merupakan *platform deployment* berbasis *cloud* yang mendukung proses otomatisasi *deployment* tanpa memerlukan konfigurasi server secara manual. Platform ini menyediakan fitur-fitur seperti integrasi dengan GitHub, *Continuous Deployment*, serta dukungan terhadap berbagai bahasa pemrograman dan layanan basis data seperti PostgreSQL dan MySQL. [15].

Railway mendukung proses *deployment* melalui berbagai metode, seperti integrasi dengan GitHub *repository*, penggunaan Railway *Command Line Interface* (CLI), maupun melalui Docker*file*. Integrasi dengan GitHub memungkinkan Railway untuk secara otomatis membangun dan mendistribusikan

aplikasi setiap kali terjadi perubahan pada *branch* tertentu dalam *repository*. Railway juga menyediakan fitur seperti *auto-scaling* dan *health check* yang berfungsi untuk memastikan setiap *instance* aplikasi berjalan dalam kondisi optimal. Kapabilitas tersebut menjadikan Railway sesuai untuk kebutuhan *deployment* aplikasi berskala kecil hingga menengah.

2.2.5 Nodemailer dan SMTP Gmail

Nodemailer adalah modul *open-source* untuk Node.js yang memungkinkan pengiriman email melalui protokol *Simple Mail Transfer Protocol* (SMTP). Modul ini mendukung pengiriman email dalam format teks, HTML, atau dengan lampiran, dan kompatibel dengan berbagai penyedia layanan email, termasuk Gmail [16].

Simple Mail Transfer Protocol atau SMTP adalah protokol standar untuk mengirimkan email antar host. Dalam SMTP, klien terhubung ke server untuk mengirimkan email dengan menyediakan alamat sumber dan tujuan, diikuti oleh isi pesan [17]. SMTP Gmail, layanan pengiriman email dari Google, memungkinkan pengiriman email melalui server smtp.gmail.com dengan autentikasi aman menggunakan kredensial pengguna atau kata sandi aplikasi (*app password)*.

2.2.6 Ultrmsg dan Axios

Ultrmsg adalah layanan API berbasis *cloud* yang memungkinkan pengiriman pesan melalui *platform* WhatsApp. Layanan ini mendukung pengiriman pesan teks, media, dan dokumen, serta otomatisasi komunikasi dengan konfigurasi sederhana melalui API.

Axios adalah klien HTTP berbasis *promise* untuk Node.js dan browser, yang mendukung pengiriman permintaan HTTP seperti GET, POST, PUT, dan DELETE. Axios menggunakan modul http bawaan Node.js di sisi server dan XMLHttpRequest di sisi browser, menyediakan cara yang konsisten untuk berinteraksi dengan API, termasuk Ultrmsg, dengan fitur seperti penanganan kesalahan dan transformasi data [18].

2.2.7 Postman

Postman adalah *platform* pengembangan dan pengujian API (*Application Programming Interface*) yang dirancang untuk membantu pengembang dalam mengelola, menguji, dan mendokumentasikan API.

Postman memungkinkan pengguna untuk mengirim permintaan ke *endpoint* API, memeriksa respons, dan mengatur alur kerja pengujian dengan fitur seperti *collections* dan *environments*. *Collections* adalah fitur inti Postman yang memungkinkan pengguna untuk mengelompokkan serangkaian permintaan API ke dalam satu wadah terorganisir. *Enviroments* merupakan kumpulan variabel yang dapat digunakan kembali dalam permintaan untuk mengatur pengujian di berbagai kondisi, seperti server lokal, *staging*, atau produksi. *Enviroments* ini memungkinkan pengguna untuk mengubah host atau otorisasi tanpa mengedit setiap permintaan secara manual.

Postman mendukung protokol komunikasi seperti HTTP, yang merupakan standar untuk pengujian API berbasis web. Kemampuan mengelola berbagai jenis permintaan, seperti GET, POST, PUT, dan DELETE, membuat Postman menjadi alat fleksibel untuk menguji fungsionalitas API dalam berbagai skenario.

Postman juga menghasilkan dokumentasi yang dapat diakses oleh tim, memastikan bahwa informasi tentang API tersedia dalam format yang jelas dan terstruktur.

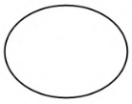
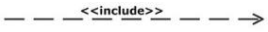
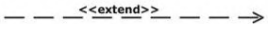
2.3 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah metode pemodelan visual yang digunakan untuk merancang sistem berorientasi objek. UML berfungsi sebagai bahasa standar untuk visualisasi, perancangan, dan pendokumentasian sistem perangkat lunak, menyediakan blueprint yang memudahkan pengembangan dan komunikasi antar pengembang [19]. UML mencakup berbagai jenis diagram, seperti *use case, activity diagram, class diagram,* dan lain-lain untuk merepresentasikan struktur dan perilaku sistem secara terstruktur.

2.3.1 *Use Case Diagram*

Penggunaan *use case diagram* dapat membantu pengembang dalam menentukan kebutuhan sistem, menjelaskan fungsi sistem kepada klien, dan mengidentifikasi aktor serta aktivitas yang didukung oleh sistem [19]. Diagram ini menggunakan simbol-simbol seperti aktor (digambarkan sebagai figur manusia), *use case* (digambarkan sebagai oval), dan garis penghubung untuk mendefinisikan hubungan antara aktor dan fungsi sistem. Berikut adalah simbol-simbol utama yang digunakan dalam *use case* diagram, sebagaimana disajikan pada Tabel 2.2.

Tabel 2.2 Simbol *Use Case Diagram*

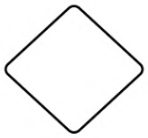
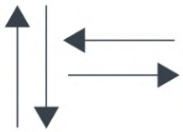
Simbol	Nama	Keterangan
	*Actor*	Entitas eksternal (*user* atau sistem lain) yang berinteraksi dengan sistem.
	*Use Case*	Fitur atau layanan yang disediakan sistem yang digunakan oleh aktor.
	Association	Garis yang menghubungkan aktor dengan *use case* (interaksi langsung).
	*Include*	Relasi yang menunjukkan satu *use case* secara wajib menyertakan *use case* lain sebagai bagian dari prosesnya.
	*Extend*	Relasi yang menunjukkan bahwa *use case* tambahan dapat dijalankan secara opsional dalam kondisi tertentu.
	*System*	Kotak besar yang membungkus semua *use case* untuk menandai batas sistem.

2.3.2 *Activity Diagram*

Activity diagram adalah salah satu jenis diagram UML yang memodelkan proses atau alur kerja dalam suatu sistem. Diagram ini menggambarkan urutan

aktivitas, keputusan, dan alur logika yang terjadi dalam sistem, sering digunakan untuk menganalisis proses bisnis atau logika aplikasi [19]. Berikut adalah simbol-simbol utama yang digunakan dalam *activity diagram*, sebagaimana disajikan pada Tabel 2.3.

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	*Start (Initial Node)*	Menandakan titik awal dari alur aktivitas.
	*Activity/Action*	Menggambarkan langkah atau aktivitas spesifik dalam proses.
	*Decision Node*	Menunjukkan titik keputusan dengan beberapa alur keluar berdasarkan kondisi.
	*Flow/Transition*	Menunjukkan alur atau transisi dari satu aktivitas ke aktivitas lain.
	*End (Final Node)*	Menandakan akhir dari alur aktivitas.

2.4 *Entity Relationship Diagram*

Diagram ERD digunakan untuk memodelkan struktur data dan hubungan antar entitas dalam sistem. ERD membantu dalam memahami bagaimana data disimpan, diakses, dan dikelola, sehingga dapat memastikan desain *database* yang efisien dan sesuai dengan kebutuhan sistem. ERD menggambarkan hubungan

antara entitas dalam sistem melalui tiga komponen utama, yaitu entitas, atribut, dan relasi [20]. Berikut adalah penjelasan dari ketiga komponen utama tersebut.

1. Entitas

Entit*as a*dalah objek atau konsep dalam dunia nyata yang memiliki keberadaan independen dan dapat diidentifikasi secara unik dalam suatu sistem. Entitas dapat berupa benda fisik (misalnya, "Mahasiswa", "Produk") atau konsep abstrak (misalnya, "Pesanan", "Kelas"). Dalam ERD, entitas digambarkan sebagai persegi panjang dengan nama entitas di dalamnya.

2. Atribut

Atribut dalam ERD merupakan karakteristik yang mendeskripsikan entitas, yang dibagi menjadi dua jenis utama: *identifier* dan *descriptor* [20]. *Identifier*, atau kunci, adalah atribut yang secara unik mengidentifikasi setiap *instance* entitas, seperti "NIM" pada entitas "Mahasiswa", yang memastikan tidak ada duplikasi data. Sebaliknya, *descriptor*, atau atribut *non-key*, memberikan informasi tambahan yang tidak unik, seperti "Nama" atau "Alamat", untuk memperkaya deskripsi entitas.

3. Relasi

Relasi dalam ERD didefinisikan berdasarkan kardinalitas, yaitu jumlah instance entitas yang dapat terhubung satu sama lain. Ada tiga jenis relasi utama berdasarkan kardinalitas, yang masing-masing memiliki karakteristik. Berikut adalah penjelasan dari ketiga jenis relasi dalam ERD.

a. *One to One*

Dalam relasi *one to one*, satu instance dari entitas A hanya dapat terkait dengan tepat satu instance dari entitas B, dan sebaliknya. Hubungan ini biasanya digunakan ketika dua entitas memiliki keterkaitan yang sangat spesifik dan eksklusif. Berikut adalah simbol visual yang digunakan untuk menggambarkan hubungan *one to one*.

Gambar 2.1 Simbol Relasi *One to One*

b. *One to Many*

Dalam relasi *one to many*, satu instance dari entitas A dapat terkait dengan banyak instance dari entitas B, tetapi satu instance dari entitas B hanya terkait dengan satu instance dari entitas A. Ini adalah jenis relasi yang paling umum dalam basis data relasional karena merefleksikan hierarki atau ketergantungan yang alami dalam banyak sistem. Berikut adalah simbol visual yang digunakan untuk menggambarkan hubungan *one to many*.

Gambar 2.2 Simbol Relasi *One to Many*

c. *Many to Many*

Dalam relasi *many to many*, banyak instance dari entitas A dapat terkait dengan banyak instance dari entitas B, dan sebaliknya. Relasi ini kompleks karena tidak dapat diimplementasikan langsung dalam basis data relasional tanpa entitas perantara (*junction table*). Berikut adalah simbol visual yang digunakan untuk menggambarkan hubungan *many to many*.

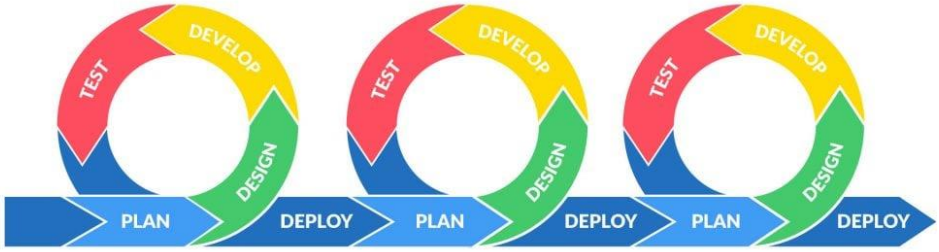
Gambar 2.3 Simbol Relasi *Many to Many*

2.5 Metode *Agile*

Metode *Agile* adalah pendekatan dalam pengembangan perangkat lunak yang mengadopsi proses iteratif dan inkremental. *Agile* menekankan iterasi yang singkat dan sering, dengan tujuan untuk menghasilkan perangkat lunak yang dapat digunakan secara cepat dan meningkatkan kolaborasi antara tim pengembang serta pengguna. Pendekatan ini berfokus pada fleksibilitas dalam merespons perubahan dibandingkan mengikuti perencanaan yang kaku.

Agile berbeda dari metode tradisional seperti *Waterfall* dalam beberapa aspek utama, yaitu;

- *Agile* lebih mengutamakan kerja sama tim dan komunikasi langsung dibandingkan prosedur formal dan dokumentasi yang berlebihan.
- Dokumentasi tetap penting, tetapi tujuan utama adalah menghasilkan perangkat lunak yang dapat digunakan sesegera mungkin.
- *Agile* mendorong keterlibatan aktif pengguna atau klien dalam setiap tahap pengembangan.
- *Agile* memungkinkan perubahan dan adaptasi selama proses pengembangan berlangsung [21].



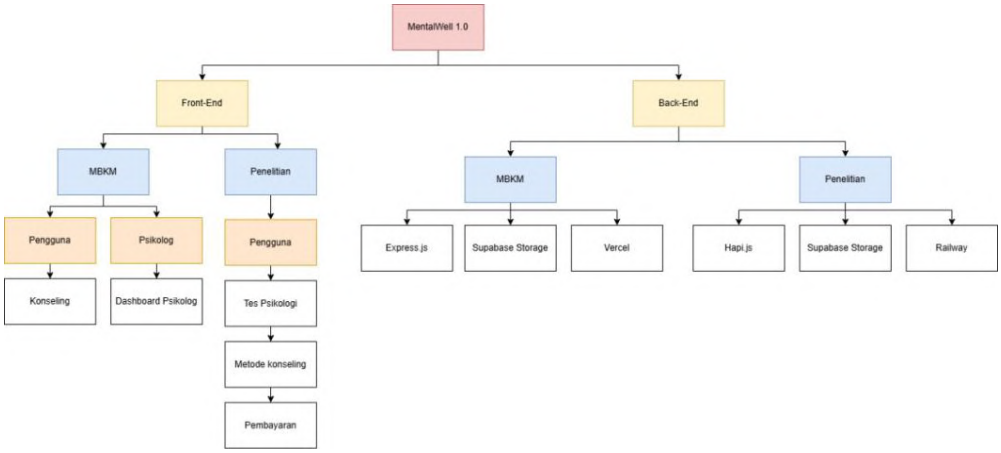
Gambar 2.4 Siklus Iteratif *Agile* [22]

Gambar di atas menunjukkan model iteratif dari metode *Agile*, di mana proses pengembangan perangkat lunak dibagi ke dalam siklus-siklus pendek yang mencakup empat tahap utama: perencanaan (*plan*), perancangan dan pengembangan (*develop*), pengujian (*test*), dan penyebaran atau implementasi (*deploy*). Setiap siklus bertujuan menghasilkan versi perangkat lunak yang dapat digunakan dan dievaluasi. Pada akhir iterasi, proses kembali ke tahap perencanaan untuk memperbaiki atau menambahkan fitur berdasarkan umpan balik pengguna. Siklus berlangsung berulang hingga produk akhir mencapai kualitas dan fungsionalitas yang diharapkan.

2.6 MentalWell 1.0

MentalWell 1.0 adalah versi pengembangan dari *capstone project* MentalWell, yang dikembangkan untuk meningkatkan aksesibilitas layanan psikologis berbasis website.

Pengembangan MentalWell 1.0 dilakukan dalam bentuk tim proyek, di mana setiap anggota memiliki peran masing-masing sesuai dengan pembagian tanggung jawab antara *back-end* dan *front-end*. Dalam pengembangan ini, MentalWell 1.0 mengalami beberapa penambahan dan perubahan signifikan dibandingkan dengan versi MentalWell sebelumnya.



Gambar 2.5 Perbandingan antara MentalWell dan MentalWell 1.0

Dari sisi *back-end*, MentalWell 1.0 masih menggunakan PostgreSQL sebagai *database* utama dan Supabase Storage untuk penyimpanan *file*, sama seperti versi sebelumnya. Struktur *database* secara keseluruhan tetap mengikuti rancangan sebelumnya, dengan beberapa penyesuaian untuk mendukung fitur baru. RESTful API yang sebelumnya dibangun menggunakan *framework* Express.js kini dikembangkan ulang dari awal menggunakan Hapi.js, sementara proses *deployment* yang sebelumnya menggunakan Vercel diganti menjadi Railway.

2.7 Penelitian Terdahulu

Dalam pengembangan *platform* layanan bantuan psikologis berbasis digital, berbagai penelitian telah dilakukan untuk mengeksplorasi teknologi serta tantangan yang dihadapi dalam implementasinya.

1. Pengembangan *Platform* Layanan Kesehatan Berbasis Web

   Beberapa penelitian telah membahas pengembangan *platform* yang memfasilitasi layanan konsultasi daring. Penelitian oleh Ilman Nawali dan Bernard Renaldy Suteja (2023) mengembangkan *platform* Parent-Care yang memungkinkan orang tua berkonsultasi dengan psikolog terkait kesehatan mental anak. *Platform* ini menggunakan Hapi.js sebagai *back-end* dengan integrasi API untuk menyimpan dan mengelola data pengguna serta sesi konsultasi [23]. Pendekatan ini menunjukkan bahwa Hapi.js dapat menjadi solusi praktis untuk membangun layanan psikologi berbasis web dengan arsitektur yang fleksibel dan mudah dikembangkan.

   Studi lain oleh Lox Xuan Wen (2022) mengembangkan *platform* berbasis web yang mengintegrasikan manajemen layanan konseling dengan fitur sosial, seperti sistem reservasi janji temu, pencatatan rekam medis, dan komunitas dukungan sosial bagi mahasiswa. Penggunaan SQL *database* dapat meningkatkan akurasi pencatatan riwayat konsultasi serta mempermudah integrasi dengan layanan lain, yang menjadi dasar penting dalam mengelola data pengguna secara terstruktur [24].

   Penelitian oleh Sadman Ahmed, Mohammad Monirujjaman Khan, Roobaea Alroobaea, dan Mehedi Masud (2023) membahas penerapan sistem reservasi janji temu berbasis web dalam layanan fisioterapi. Studi ini menekankan bahwa konsep reservasi daring dapat meningkatkan efisiensi dan aksesibilitas layanan kesehatan, yang juga relevan untuk diterapkan dalam sistem konsultasi psikologi untuk mempermudah pengguna mengakses layanan tanpa hambatan geografis [25].

2. Pengelolaan Database

   Pengelolaan *database* yang efisien menjadi aspek penting dalam mendukung performa *platform* berbasis web. Penelitian oleh Umi Zahroh, Rachmadita Andreswari, Ekky Novriza Alam (2023) menunjukkan bahwa

penggunaan PostgreSQL sebagai *database* utama dapat mendukung pengelolaan berbagai skema data, termasuk data pengguna, transaksi, dan sesi konsultasi. Pendekatan ini memungkinkan data tersimpan secara terstruktur dan mudah diakses untuk kebutuhan operasional sistem [26].

Penelitian oleh Saputra, Kurniawan, dan Rahayu (2023) menyoroti pentingnya perancangan *Entity Relationship Diagram* (ERD) untuk mengoptimalkan penyimpanan dan pengelolaan data. Studi ini merancang 9 tabel utama dengan hubungan *one-to-many* dan *foreign key*, yang membantu memastikan integritas data serta mempermudah pengelolaan entitas yang kompleks, seperti pengguna, psikolog, dan sesi konsultasi.

Penelitian oleh Nejković, Marković, dan Petrović (2024) membahas pemanfaatan Supabase untuk meningkatkan efisiensi manajemen *database* dan sinkronisasi data secara *realtime*. Studi ini menunjukkan bahwa Supabase dapat menyederhanakan proses pengelolaan data tanpa mengorbankan performa sistem, menjadikannya pilihan yang relevan untuk *platform* yang membutuhkan konektivitas data yang cepat dan stabil [27].

3. Keamanan dan Autentikasi Sistem

Keamanan sistem menjadi prioritas dalam pengembangan layanan yang menangani data sensitif pengguna. Penelitian oleh Siyu Wang dan Haishan Wang (2023) menunjukkan bahwa pemisahan antara *back-end* dan *front-end* dapat meningkatkan efisiensi pengembangan serta mempermudah pemeliharaan sistem. Studi ini juga menyoroti pentingnya penggunaan protokol keamanan seperti OAuth dan JWT untuk memperkuat autentikasi, melindungi data sensitif, dan mencegah akses yang tidak sah [28].

Penelitian lain oleh Umi Zahroh, Rachmadita Andreswari, Ekky Novriza Alam (2022) menunjukkan bahwa penggunaan JWT (JSON Web Token) dapat meningkatkan keamanan akses pengguna. JWT memungkinkan proses autentikasi yang lebih aman dengan memberikan token yang berisi klaim pengguna, yang kemudian diverifikasi setiap kali pengguna mengakses fitur-fitur sensitif dalam aplikasi. Hal ini memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses data tertentu, sehingga memperkuat perlindungan privasi dan keamanan informasi pengguna [26].

4. Pengembangan Sistem Informasi Berbasis Web dengan Metode Agile

   Penelitian oleh Rahman, Hastuti, dan Lestari (2024) menunjukkan bahwa metode Agile dapat mempercepat dan merapikan proses pengembangan sistem. Pengembangan dilakukan dalam empat iterasi selama delapan minggu, meliputi tahapan perencanaan, desain, pengembangan, pengujian, dan penyebaran [22].

   Hasil penelitian ini menunjukkan bahwa pendekatan Agile membuat tim lebih adaptif terhadap perubahan kebutuhan pengguna dan mempercepat pengiriman fitur yang stabil. Temuan ini relevan untuk pengembangan *platform* konsultasi psikologi yang memerlukan pembaruan berkelanjutan sesuai kebutuhan pengguna.

Pengembangan MentalWell 1.0 mengacu pada aspek penting yang mendukung keberhasilan *platform* digital, sebagaimana dibuktikan dalam berbagai penelitian terdahulu. Penggunaan Hapi.js sebagai *back-end*, PostgreSQL untuk pengelolaan data, dan JWT untuk autentikasi membantu membangun sistem yang terstruktur. Teknologi lain seperti Socket.io dan WebSocket dapat mendukung komunikasi *realtime*, sementara ERD dan Supabase mempermudah pengelolaan data.

Metode *Agile* digunakan dalam penelitian ini karena memungkinkan tim bekerja secara bertahap dan fleksibel terhadap perubahan kebutuhan pengguna. Proses yang berulang memungkinkan fitur-fitur dikembangkan, diuji, dan diperbaiki secara berkala, sehingga MentalWell 1.0 terus berkembang dan memberikan layanan lebih baik.

BAB III
METODOLOGI PENELITIAN

3.1 Waktu dan Tempat

3.1.1 Waktu Penelitian

Penelitian ini dilakukan dalam beberapa tahapan yang mencakup *Planning, Design, Development, Testing, Documentation,* dan *Deployment*. Setiap tahapan memiliki rentang waktu yang telah ditentukan agar proses penelitian berjalan secara sistematis dan terstruktur. Tabel berikut menunjukkan distribusi waktu untuk masing-masing tahapan penelitian dari Januari hingga Juli.

Tabel 3.1 Waktu Penelitian

No.	Tahapan Penelitian	Jan	Feb	Mar	Apr	Mei	Jun	Jul
1.	*Planning*							
2.	*Design*							
3.	*Development*							
4.	*Testing*							
5.	*Documentation*							
6.	*Deployment*							

3.1.2 Tempat Penelitian

Penelitian ini dilaksanakan di Jurusan Teknik Elektro, Universitas Lampung, yang berlokasi di Jalan Prof. Dr. Ir. Sumantri Brojonegoro No. 1, Kel. Gedong Meneng, Kec. Rajabasa, Kota Bandar Lampung, Prov. Lampung.

3.2 Alat dan Bahan Penelitian

3.2.1 Alat Penelitian

Penelitian ini menggunakan berbagai alat bantu yang mendukung proses pengembangan sistem MentalWell 1.0. Rincian alat penelitian dikelompokkan ke dalam tiga kategori: perangkat keras, perangkat lunak umum, dan paket dependensi yang digunakan dalam pengembangan *back-end*.

Tabel 3.2 Perangkat Keras

No.	Alat	Fungsi
1.	Macbook Air M1 (2020) • Operating System (OS) : MacOS Sequoia v.15.5 • System Architecture : arm64 • Processor : Apple M1 • Memory (RAM): 8GB	Digunakan untuk *coding*, desain, dan pengujian

Tabel 3.3 Perangkat Lunak

No.	Alat	Fungsi
1.	Node.js	*Runtime environment* untuk menjalankan kode JavaScript di sisi server.
2.	JavaScript	Bahasa pemrograman utama untuk mengembangkan API aplikasi.
3.	Visual Studio Code v.1.92.2	*Text editor* untuk menulis kode program.
4.	Hapi.js	*Framework back-end* untuk membangun API.
5.	Dbdiagram.io	Digunakan untuk merancang *database* dalam bentuk *Entity Relationship Diagram* (ERD).
6.	Lucidchart	Digunakan untuk membuat *Activity Diagram*.
7.	Supabase (Database + Storage)	Digunakan sebagai *back-end-as-a-service* (BaaS) untuk menyediakan *database* PostgreSQL dan penyimpanan *file* berbasis *cloud*.
8.	PostgreSQL (Melalui Supabase)	Digunakan sebagai sistem manajemen basis data relasional (RDBMS) untuk menyimpan dan mengelola data secara terstruktur.

Tabel 3.3 (Lanjutan)

No.	Alat	Fungsi
9.	Storage (Melalui Supabase)	Digunakan untuk menyimpan dan mengelola media yang terintegrasi dalam website.
10.	Server SMTP Gmail	Digunakan sebagai server eksternal untuk mengirimkan email notifikasi dari aplikasi melalui akun Gmail.
11.	Ultrmsg	Layanan API pihak ketiga untuk mengirimkan pesan WhatsApp secara otomatis.
12.	GitHub	Digunakan untuk menyimpan kode dan juga proses *deployment*.
13.	Postman	Digunakan untuk menguji dan melakukan dokumentasi API

Tabel 3.4 Paket Dependensi

No.	Nama Paket	Versi	Fungsi Utama
1.	@hapi/hapi	^21.3.12	*Framework* utama untuk membangun server dan manajemen rute.
2.	joi	^17.1.1	Validasi data berbasis skema.
3.	@supabase/supabase-js	^2.48.1	SDK untuk koneksi ke layanan Supabase.
4.	axios	^1.9.0	Digunakan untuk melakukan permintaan HTTP ke layanan pihak ketiga, seperti pengiriman pesan WhatsApp melalui Ultramsg API.
5.	bcryptjs	^3.0.2	*Hashing* dan verifikasi kata sandi.
6.	dayjs	^1.11.13	Manipulasi dan format tanggal.
7.	dotenv	^16.4.7	Memuat variabel lingkungan dari *file* .env.
8.	jsonwebtoken	^9.0.2	Pembuatan dan verifikasi JSON Web Token (JWT) untuk autentikasi.

Tabel 3.4 (Lanjutan)

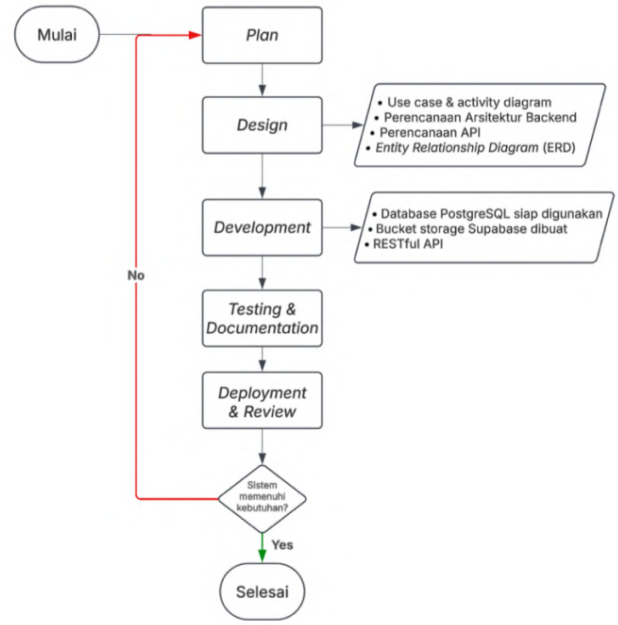
No.	Nama Paket	Versi	Fungsi Utama
9.	nanoid	^3.3.9	Pembuatan id unik pendek dan aman.
10.	nodemailer	^6.10.1	Pengiriman email notifikasi.
11.	nodemon	^3.1.9	Menjalankan server dengan otomatisasi *reload* saat terjadi perubahan kode.

3.2.2 Bahan Penelitian

Bahan penelitian ini diperoleh melalui studi literatur yang mencakup berbagai referensi, seperti jurnal ilmiah, buku, serta dokumentasi atau artikel teknis resmi yang relevan dengan topik yang dikaji.

3.3 Tahapan Penelitian

Penelitian ini dilakukan dengan menggunakan metode *Agile* karena pendekatannya yang iteratif, fleksibel, dan kolaboratif. *Agile* dipilih karena memungkinkan pengembangan bertahap, responsif terhadap perubahan, serta mendukung komunikasi yang lebih efektif dalam tim.



Gambar 3.1 Tahapan Penelitian

3.3.1 *Plan*

Tahap awal pengembangan dilakukan dengan mengidentifikasi kebutuhan sistem berdasarkan studi literatur dan diskusi teknis bersama tim *front-end*. Studi literatur digunakan untuk memahami pendekatan dan teknologi yang sesuai, seperti penggunaan arsitektur REST API, JSON Web Token (JWT) untuk autentikasi, serta pemanfaatan layanan Supabase sebagai *platform backend-as-a-service*.

Di sisi lain, diskusi dengan tim *front-end* dilakukan untuk menyesuaikan kebutuhan implementasi dari sisi antarmuka pengguna, termasuk skenario penggunaan dan alur interaksi sistem.

Hasil dari tahap ini dirumuskan menjadi kebutuhan awal sistem, yang dikategorikan ke dalam kebutuhan fungsional dan non-fungsional

3.2.1.1 Studi Literatur

Studi literatur pada penelitian ini telah dibahas pada bab 2. Pembahasan mencakup teknologi dan konsep yang digunakan dalam pengembangan *back-end* sistem, antara lain prinsip REST API sebagai arsitektur komunikasi antar sistem, JSON Web Token (JWT) untuk autentikasi dan otorisasi, serta *framework* Hapi.js sebagai dasar pengembangan server. Sistem ini juga menggunakan layanan Supabase dengan PostgreSQL sebagai *database* utama, serta fitur Supabase Storage untuk pengelolaan *file* dan Supabase Realtime untuk mendukung komunikasi *realtime*. Proses *deployment* dilakukan menggunakan Railway sebagai *platform hosting*. Penelitian ini juga memanfaatkan alat bantu seperti UML untuk perancangan alur sistem, ERD untuk perancangan struktur *database*, serta integrasi dengan layanan pihak ketiga seperti UltraMsg untuk pengiriman pesan WhatsApp dan SMTP server untuk email notifikasi.

3.2.1.2 Spesifikasi Kebutuhan

Spesifikasi kebutuhan pada sistem ini dibagi menjadi dua, yaitu kebutuhan fungsional dan kebutuhan non-fungsional. Kebutuhan fungsional menjelaskan fitur-fitur utama yang harus disediakan oleh sistem, sedangkan kebutuhan non-fungsional berfokus pada bagaimana sistem *back-end* bekerja dari sisi teknis.

Tabel 3.5 Kebutuhan Fungsional

Kode Fungsional	Deskripsi
KF-01	Sistem menyediakan fitur layanan konseling dengan profesional.
KF-02	Sistem menyediakan fitur tes psikologi untuk memahami kondisi mental.
KF-03	Sistem mendukung fitur *chat* yang digunakan pada sesi konseling.
KF-04	Sistem menyediakan fitur konfirmasi pembayaran pada layanan daftar konseling

Tabel 3.6 Kebutuhan Non-Fungsional

Kode Non-Fungsional	Deskripsi
KNF-01	Sistem menggunakan JSON Web Token untuk autentikasi dan otorisasi pengguna.
KNF-02	Semua pertukaran data dilakukan dalam format JSON.
KNF-03	Database didesain relasional untuk menjaga konsistensi data.
KNF-04	Sistem dirancang dengan struktur modular.
KNF-05	Sistem menjalankan tugas terjadwal *(scheduled task)* untuk menjalankan pembaruan status sesi konseling secara berkala tanpa input manual.
KNF-06	Sistem di*deploy* ke *platform cloud* dan dapat diakses oleh aplikasi *front-end* melalui internet
KNF-07	Setiap *endpoint back-end* didokumentasikan menggunakan Postman untuk memudahkan proses integrasi dan pengujian.

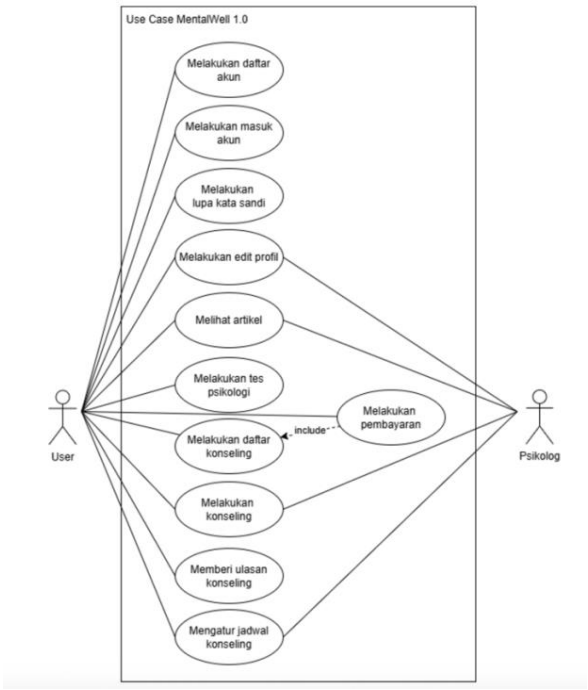
3.3.2 *Design*

Tahapan ini membahas rancangan sistem yang dikembangkan untuk aplikasi MentalWell 1.0. Perancangan dilakukan untuk memberikan gambaran menyeluruh mengenai alur kerja, struktur data, serta interaksi komponen di dalam sistem sebelum tahap implementasi dilakukan.

Perancangan diawali dengan penyusunan *use case diagram* untuk menggambarkan hubungan antara aktor dan fungsionalitas sistem. Activity diagram, selanjutnya disajikan untuk memperjelas alur proses utama yang terjadi pada sistem. *Entity Relationship Diagram* (ERD) digunakan untuk merepresentasikan struktur basis data beserta relasinya. Arsitektur *back-end* dijabarkan untuk menunjukkan koneksi aplikasi dengan layanan *database* serta pengaturan aliran data. Perencanaan *endpoint* API disusun sebagai acuan pengembangan layanan back-end agar setiap fitur dapat diakses sesuai kebutuhan.

3.3.2.1 *Use Case Diagram*

Use case diagram MentalWell 1.0 dapat dilihat pada Gambar 3.2 berikut.



Gambar 3.2 *Use Case Diagram* MentalWell 1.0

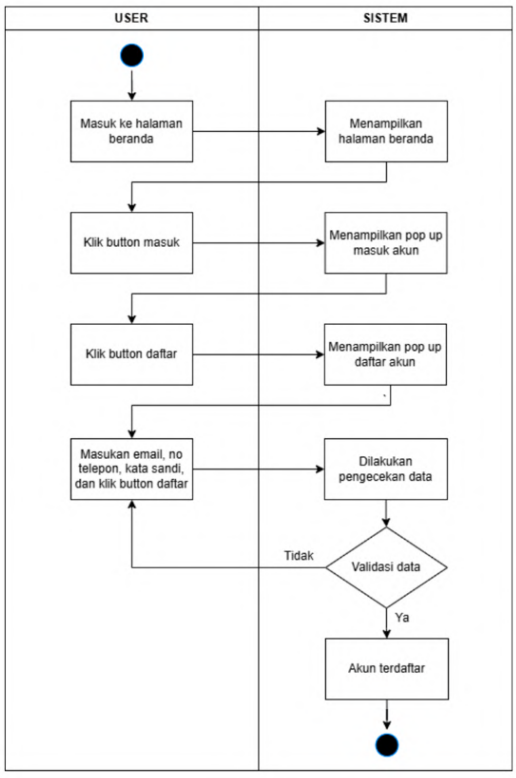
Use case diagram pada Gambar 3.2 menggambarkan interaksi antara aktor dengan sistem MentalWell 1.0. Terdapat dua aktor utama, yaitu *User* dan Psikolog. *User* merupakan pengguna aplikasi yang dapat melakukan berbagai aktivitas seperti mendaftar akun, masuk ke sistem, mengatur profil, melihat artikel, melakukan tes psikologi, mendaftar konseling, hingga memberi ulasan terhadap sesi konseling yang diikuti.

Psikolog sebagai aktor kedua berperan dalam memberikan layanan konseling, mengatur jadwal, serta menerima pembayaran dari sesi konseling yang dipesan *User*. Proses pembayaran menjadi bagian terintegrasi (*include*) dari pendaftaran konseling.

3.2.2.2 *Activity Diagram*

Activity diagram menjelaskan alur aktivitas utama yang terjadi saat pengguna berinteraksi dengan sistem MentalWell 1.0. Diagram ini disusun dari perspektif *front-end*, dengan fokus pada interaksi antara pengguna dan antarmuka sistem, serta tanggapan sistem yang bersifat fungsional dan visual. Setiap diagram menggambarkan proses yang terjadi di sisi pengguna dan bagaimana sistem merespons input tersebut melalui antarmuka, tanpa menggambarkan proses *back-end* secara mendetail.

A. *Activity Diagram* Daftar Akun



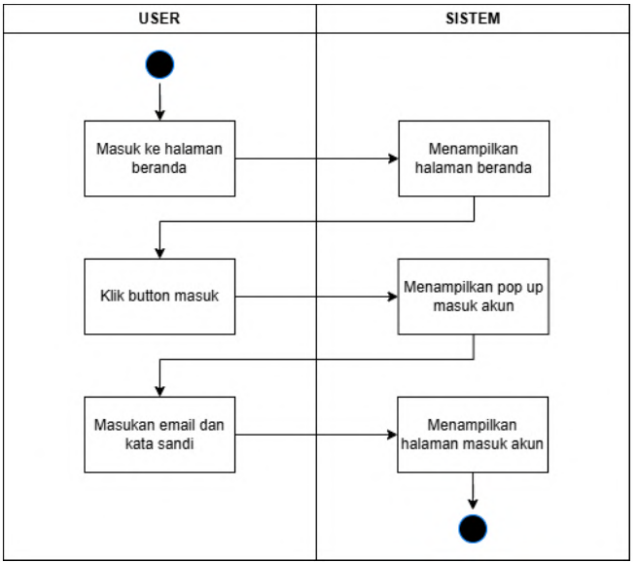
Gambar 3.3 *Activity Diagram* Daftar Akun

Proses pendaftaran akun pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.3, diawali ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda kepada *user*. *User* selanjutnya menekan tombol "masuk" dan sistem menampilkan *pop up* masuk akun.

Jika *user* belum memiliki akun, *user* memilih opsi "daftar" sehingga sistem menampilkan pop up daftar akun. Pada tahap ini *user* mengisi data pendaftaran berupa email, nomor telepon, kata sandi, lalu menekan tombol daftar. Sistem akan melakukan pengecekan data untuk memvalidasi kebenaran dan kelengkapan informasi.

Apabila data yang diinput tidak valid, sistem mengirimkan informasi bahwa pendaftaran gagal dan meminta *user* memperbaiki data yang dimasukkan. Namun jika data valid, sistem melanjutkan proses dan mendaftarkan akun *user* ke dalam *database*. Proses pendaftaran kemudian berakhir dengan status akun telah terdaftar.

B. *Activity Diagram* Masuk Akun

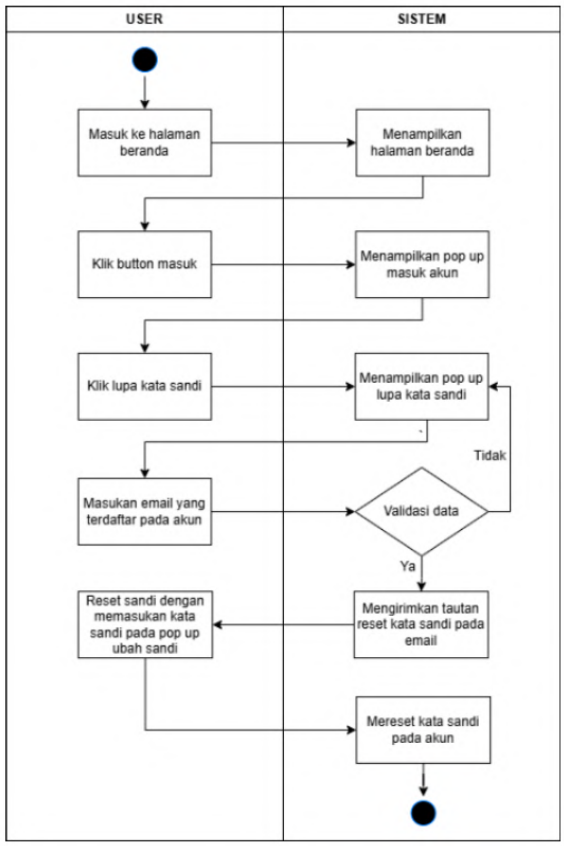


Gambar 3.4 *Activity Diagram* Masuk Akun

Proses masuk akun pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.4, diawali ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda. *User* selanjutnya menekan tombol "masuk", dan sistem merespons dengan menampilkan *pop up* masuk akun.

Pada tahap berikutnya, *user* memasukkan email dan kata sandi. Sistem memproses data tersebut dan menampilkan halaman masuk akun, menandakan bahwa proses login telah selesai dan *user* berhasil masuk ke dalam sistem.

C. *Activity Diagram* Lupa Kata Sandi



Gambar 3.5 *Activity Diagram* Lupa Kata Sandi

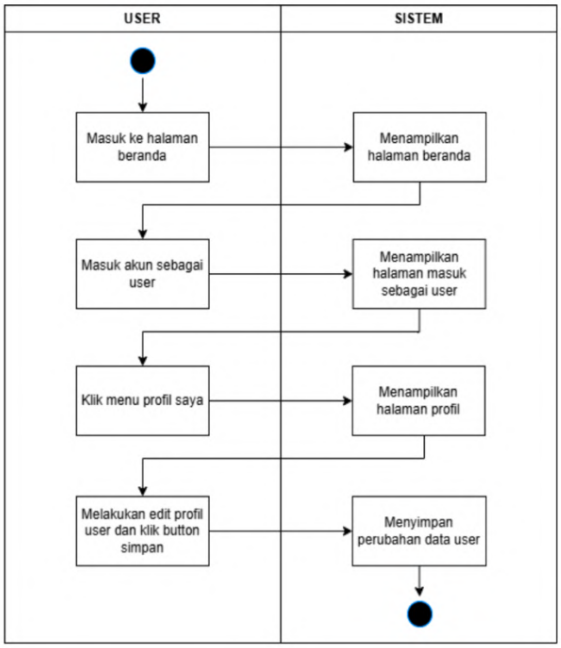
Proses lupa kata sandi pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.5, dimulai ketika *user* membuka halaman beranda. Sistem kemudian menampilkan halaman tersebut dan *user* menekan tombol "masuk" sehingga sistem memunculkan pop up masuk akun.

Jika *user* tidak ingat kata sandi, mereka menekan tombol "lupa kata sandi" dan sistem menampilkan pop up lupa kata sandi. *User* kemudian memasukkan email yang terdaftar pada sistem. Sistem akan melakukan validasi data:

- Jika email tidak terdaftar, proses berhenti dan sistem tidak mengirimkan tautan reset.
- Jika email valid, sistem mengirimkan tautan reset kata sandi ke email *user*.

User dapat mengatur ulang kata sandi dengan memasukkan kata sandi baru melalui *pop-up* yang tersedia. Sistem kemudian melakukan proses *reset* kata sandi pada akun, sehingga kata sandi *user* berhasil diperbarui.

D. *Activity Diagram* Edit Profil

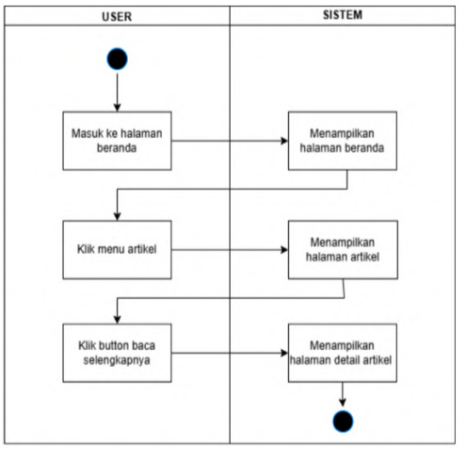


Gambar 3.6 *Activity Diagram* Edit Profil

Proses *edit* profil pada sistem MentalWell 1.0 ditunjukkan pada Gambar 3.6. Alur dimulai ketika pengguna mengakses halaman beranda, kemudian sistem menampilkan halaman tersebut. Pengguna melakukan login sebagai *user*, dan sistem menampilkan halaman masuk akun sesuai peran yang digunakan.

User kemudian memilih menu “Profil Saya” dan sistem akan menampilkan halaman profil. Pada tahap ini, *user* dapat melakukan perubahan data dengan memilih opsi edit profil, mengisi informasi yang ingin diperbarui, lalu menekan tombol simpan. Sistem merespons dengan menyimpan perubahan data *user* ke dalam basis data. Proses edit profil pun selesai.

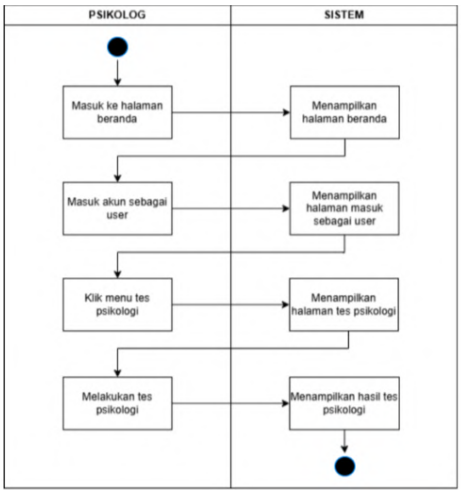
E. *Activity Diagram* Melihat Artikel



Gambar 3.7 *Activity Diagram* Melihat Artikel

Proses melihat artikel pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.7, dimulai ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda, lalu *user* memilih menu artikel dan sistem menampilkan halaman artikel. Sistem akan memunculkan halaman detail artikel yang menampilkan isi artikel secara lengkap setelah *user* menekan tombol "baca selengkapnya" pada artikel yang diinginkan.

F. *Activity Diagram* Tes Psikologi

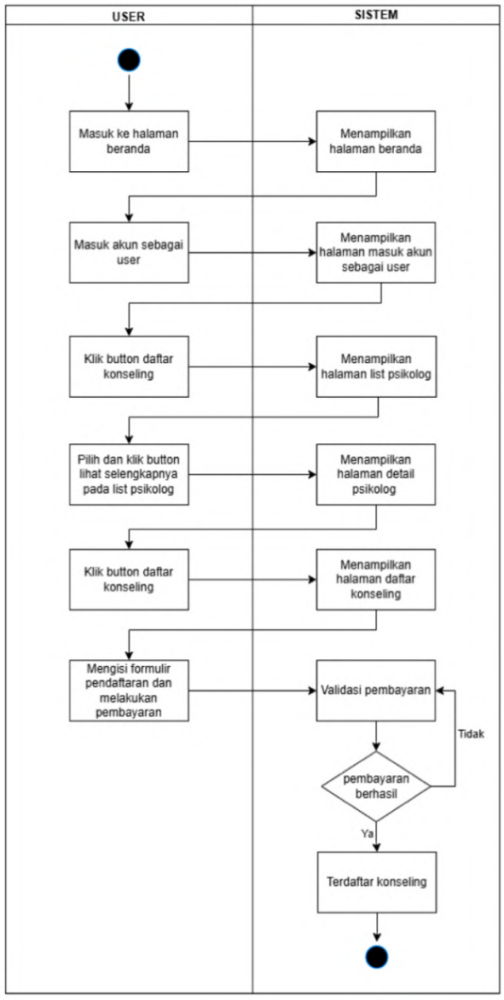


Gambar 3.8 *Activity Diagram* Tes Psikologi

Proses tes psikologi pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.8, dimulai ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda. *User* lalu melakukan login sebagai *user*, sehingga sistem menampilkan halaman masuk akun sebagai *user*.

User yang telah berhasil login selanjutnya memilih menu tes psikologi, yang kemudian ditampilkan oleh sistem. *User* kemudian melakukan pengisian dan penyelesaian tes tersebut, dan sistem akan menampilkan hasil tes psikologi sebagai *output* dari proses ini.

G. *Activity Diagram* Daftar Konseling



Gambar 3.9 *Activity Diagram* Daftar Konseling

Proses pendaftaran konseling pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.9, dimulai ketika *user* mengakses halaman beranda dan sistem menampilkan halaman beranda. *User* selanjutnya melakukan login sebagai *user* sehingga sistem menampilkan halaman masuk akun sebagai *user*.

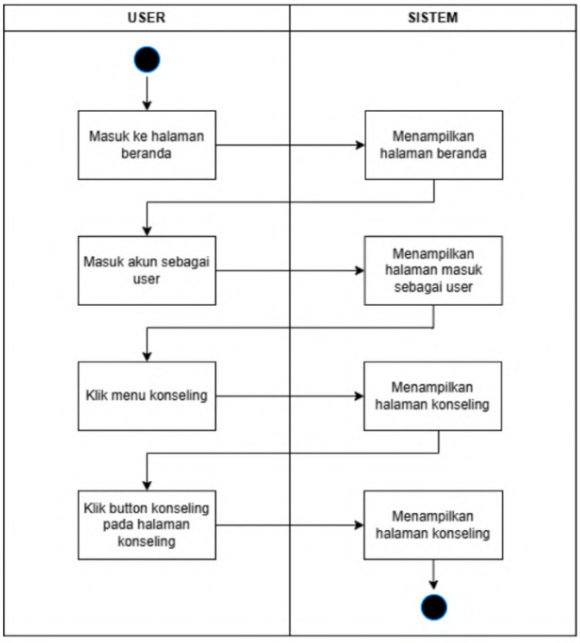
Login yang berhasil memungkinkan *user* memilih menu daftar konseling. Pada tahap ini, *user* memilih psikolog dan klik tombol daftar konseling, lalu sistem menampilkan halaman jadwal konseling.

User kemudian mengisi formulir pendaftaran dan mengunggah bukti pembayaran. Sistem akan melakukan validasi pembayaran:

- Jika pembayaran tidak valid, proses pendaftaran konseling tidak dapat dilanjutkan.
- Jika pembayaran valid, sistem mengonfirmasi bahwa *user* terdaftar pada konseling.

Proses pendaftaran konseling selesai setelah validasi berhasil.

H. *Activity Diagram* Konseling

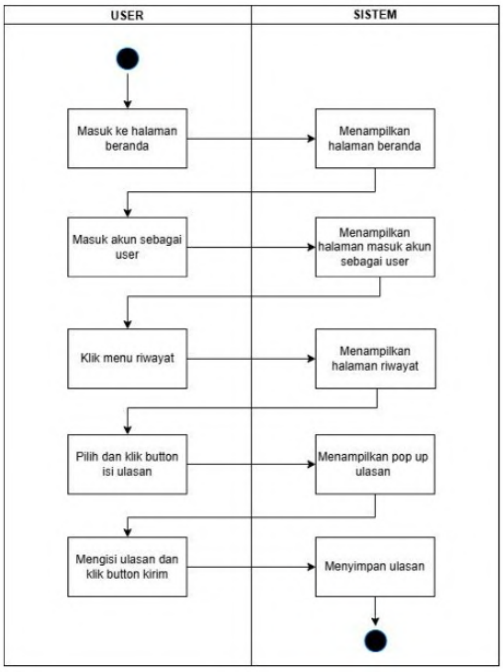


Gambar 3.10 *Activity Diagram* Konseling

Proses konseling pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.10, diawali ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda kepada *user*. Proses dilanjutkan dengan login ke dalam sistem sebagai *user*, lalu sistem menampilkan halaman masuk akun *user*.

User yang telah berhasil login dapat memilih menu konseling yang tersedia, lalu sistem menampilkan halaman konseling. Di halaman tersebut, *user* menekan tombol *chat* untuk memulai proses konseling.

I. *Activity Diagram* Ulasan Konseling



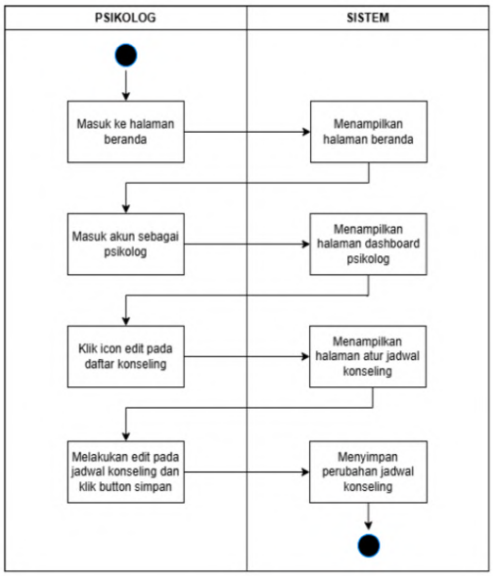
Gambar 3.11 *Activity Diagram* Ulasan Konseling

Proses ulasan konseling pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.11, diawali ketika *user* mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda. *User* kemudian melakukan login dan sistem akan menampilkan halaman masuk akun *user*.

Login yang berhasil memungkinkan *user* memilih menu riwayat untuk melihat sesi konseling sebelumnya, dan sistem menampilkan halaman riwayat. *User* selanjutnya memilih dan mengklik tombol untuk mengisi ulasan, yang memicu sistem menampilkan *pop up* ulasan.

Pada tahap akhir, *user* mengisi ulasan sesuai pengalaman yang didapatkan dan menekan tombol kirim. Sistem kemudian menyimpan ulasan tersebut sebagai *output* dari proses ini.

J. *Activity Diagram* Atur Jadwal Konseling

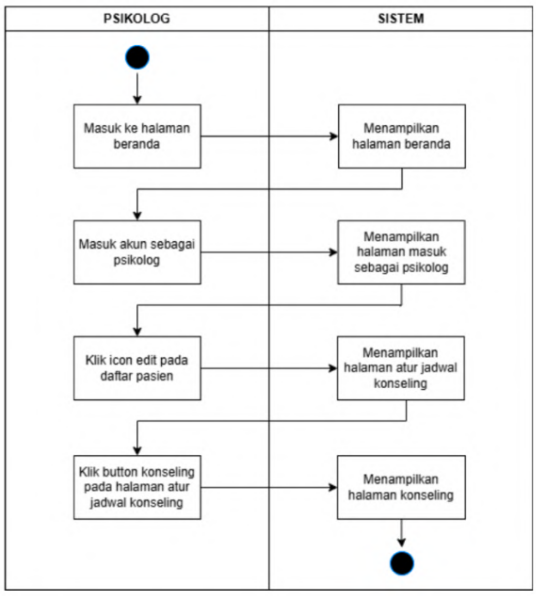


Gambar 3.12 *Activity Diagram* Atur Jadwal Konseling

Proses pengaturan jadwal konseling pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.12, dimulai ketika psikolog mengakses halaman beranda, dan sistem menampilkan halaman tersebut. Psikolog kemudian melakukan login sebagai akun psikolog, sehingga sistem menampilkan halaman dashboard psikolog. Dari *dashboard*, psikolog mengklik ikon *edit* pada daftar konseling untuk melakukan pengaturan jadwal, yang membuat sistem menampilkan halaman atur jadwal konseling.

Pada tahap ini, psikolog dapat melakukan perubahan pada jadwal konseling yang tersedia dan mengklik tombol simpan. Sistem akan menyimpan perubahan tersebut sebagai hasil akhir dari proses ini.

K. *Activity Diagram* Konseling Psikolog



Gambar 3.13 *Activity Diagram* Konseling Psikolog

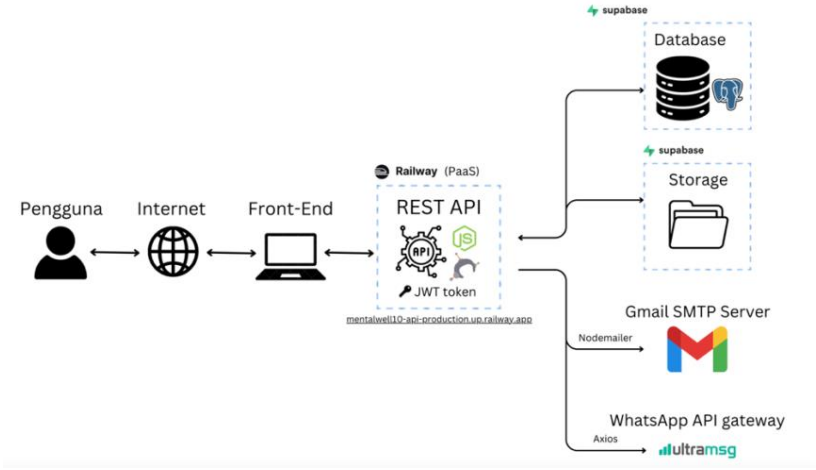
Proses konseling yang dilakukan oleh psikolog pada sistem MentalWell 1.0, seperti yang ditunjukkan pada Gambar 3.13, dimulai ketika psikolog mengakses halaman beranda. Sistem kemudian menampilkan halaman beranda. Psikolog selanjutnya melakukan login sebagai akun psikolog, dan sistem akan menampilkan halaman masuk sebagai psikolog.

Psikolog yang telah berhasil masuk dapat mengedit jadwal konseling. Aksi ini dilakukan dengan mengklik ikon *edit* pada daftar pasien, sehingga sistem menampilkan halaman atur jadwal konseling. Di halaman tersebut, psikolog menekan tombol konseling untuk memulai sesi dan proses diakhiri dengan ditampilkannya halaman konseling.

3.2.2.3 Perencanaan Arsitektur *Back-end*

MentalWell 1.0 dirancang dengan arsitektur *monolithic*, mengintegrasikan semua fitur *back-end* dalam satu aplikasi tunggal menggunakan *framework* Hapi.js. Pendekatan ini dipilih untuk mempermudah pengembangan awal, pengelolaan kode, dan integrasi fitur. Berikut adalah komponen utama arsitektur *back-end* untuk MentalWell 1.0.

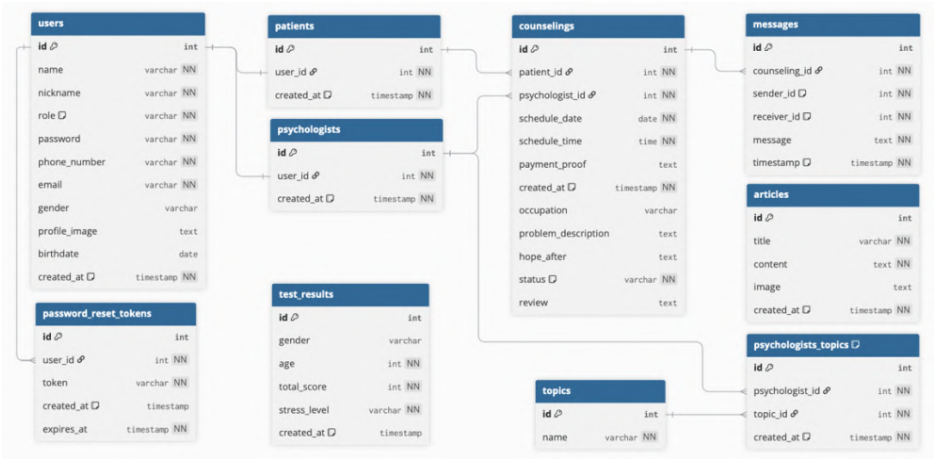
1.	*Framework Back-end*	:	Menggunakan Hapi.js untuk membangun API yang modular, terstruktur, dan mudah dikembangkan.
2.	*Database*	:	Menggunakan PostgreSQL yang disediakan oleh Supabase untuk menyimpan data pengguna, informasi konseling, serta data percakapan. Fitur percakapan yang bersifat *realtime* pada MentalWell 1.0 didukung oleh Supabase Realtime yang diakses langsung dari sisi *front-end*.
3.	Penyimpanan *File*	:	Supabase Storage digunakan untuk menyimpan *file* seperti bukti pembayaran dan foto profil.
4.	Autentikasi & Otorisasi	:	Menggunakan JWT dengan skema *Role-Based Access Control* (RBAC) untuk mengatur akses berdasarkan peran, yaitu Administrator, Psikolog, dan Pasien.
5.	Notifikasi	:	Pengiriman notifikasi melalui Nodemailer (email) dan Axios yang terintegrasi dengan Ultramsg untuk pengiriman WhatsApp, dijalankan dari *back-end* API.
7.	*Deployment*	:	Railway digunakan sebagai *platform* untuk proses *build* dan *deploy* otomatis (*Continuous Integration/Continuous Deployment*, CI/CD) API *back-end*.



Gambar 3.14 Arsitektur *Back-end* MentalWell 1.0

3.2.2.4 *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram (ERD) digunakan untuk memodelkan hubungan antarentitas dalam basis data secara terstruktur dan sistematis. Diagram ini menyajikan komponen data yang digunakan dalam sistem, mulai dari entitas, atribut, hingga relasi antarentitas, guna memastikan bahwa rancangan basis data dapat mendukung seluruh proses bisnis yang berjalan.



Gambar 3.15 *Entity Relationship Diagram* (ERD) MentalWell 1.0

Gambar 3.15 menunjukkan tiga entitas utama sistem, yaitu *users*, *patients*, dan *psychologists*. Tabel *users* menyimpan data akun dan profil umum seluruh pengguna yang terdaftar. Peran pengguna kemudian dipisahkan ke dalam tabel

patients dan *psychologists* untuk menyimpan data spesifik sesuai jenis peran masing-masing.

Pencatatan aktivitas utama dalam sistem dilakukan melalui tabel *counselings* yang menyimpan data sesi konseling antara pasien dan psikolog. Setiap sesi konseling memiliki riwayat percakapan yang tersimpan di tabel *messages*.

Sistem juga dilengkapi dengan sejumlah fitur pendukung. Tabel *articles* menyimpan konten artikel yang dapat diakses pengguna. Fitur pencarian psikolog berdasarkan bidang keahlian didukung oleh tabel *topics* dan *psychologists_topics*. Dan tabel *tes_result* untuk menyimpan hasil tes psikologis pasien.

3.2.2.5 Perencanaan *Endpoint*

API MentalWell 1.0 dirancang sebagai RESTful API untuk mendukung operasi CRUD dan fitur konsultasi psikologi daring. API ini diimplementasikan dengan *Role-Based Access Control* (RBAC) menggunakan JWT untuk mengamankan akses berdasarkan peran (Publik, Administrator, Psikolog, Pasien), dan di*deploy* di Railway. Tabel-tabel di bawah ini menyajikan daftar *endpoint* yang dikelompokkan sesuai peran pengguna.

Tabel 3.7 Daftar *Endpoint* Publik

No.	*Endpoint*	*Method*	Deskripsi
1.	*/register*	*POST*	Registrasi akun untuk pasien
2.	*/login*	*POST*	Login dan mendapatkan token JWT
3.	*/forgot-password*	*POST*	Permintaan reset password
4.	*/reset-password*	*POST*	Reset password dengan token
5.	*/articles*	*GET*	Melihat daftar artikel yang tersedia
6.	*/articles/:id*	*GET*	Melihat detail artikel yang dipilih

Tabel 3.8 Daftar *Endpoint* Pasien

No.	*Endpoint*	*Method*	Deskripsi
1.	*/profile*	*GET*	Melihat profil
2.	*/profile*	*PUT*	Mengedit profil
3.	*/my-data*	*GET*	Melihat data pribadi untuk autofill dalam proses daftar konseling
4.	*/psychologists*	*GET*	Melihat daftar psikolog
5.	*/psychologists/:id*	*GET*	Melihat detail psikolog yang dipilih
6.	*/psychologists/search*	*GET*	Melihat daftar psikolog yang dicari
7.	*/counseling/:id*	*POST*	Melakukan pemesanan sesi konseling dengan psikolog yang dipilih
8.	*/counselings*	*GET*	Melihat daftar riwayat konseling
9.	*/counselings/:id*	*GET*	Melihat detail konseling yang dipilih
10.	*/counseling/:id/review*	*POST*	Menambahkan ulasan untuk psikolog setelah sesi konseling

Tabel 3.9 Daftar *Endpoint* Psikolog

No.	*Endpoint*	*Method*	Deskripsi
1.	*/psychologist/profile*	*GET*	Melihat profil
2.	*/psychologist/ availability*	*PUT*	Mengedit ketersediaan terkini
3.	*/psychologist/ counselings*	*GET*	Melihat daftar konseling
4.	*/psychologist/ counselings/:id*	*GET*	Melihat detail konseling yang dipilih
5.	*/psychologist/ counselings/:id/status*	*PUT*	Mengubah status sesi konseling menjadi ‘selesai’

3.3.3 *Development*

Tahap *development* mencakup proses pembuatan REST API untuk mendukung fungsionalitas aplikasi. Proses ini meliputi pembuatan *endpoint* API, penerapan logika bisnis, serta integrasi dengan *database*.

3.3.4 *Testing & Documentation*

Tahap *development* kemudian diikuti dengan pengujian API (API testing) menggunakan Postman untuk memastikan API berjalan sesuai dengan spesifikasi. Pengujian mencakup validasi setiap *endpoint* serta verifikasi API yang telah di-*deploy*. Postman juga dimanfaatkan untuk menyusun dokumentasi API agar dapat digunakan oleh tim *front-end*.

3.3.5 *Deployment & Review*

Tahap *deployment* mencakup *handover* API ke tim *front-end*, penyusunan dokumentasi API, serta integrasi API dalam sistem. Pemantauan *error* dan perbaikan *bug* dilakukan berdasarkan *feedback* dari tim *front-end* untuk memastikan API berfungsi sesuai kebutuhan.

BAB V
PENUTUP

5.1 Kesimpulan

Hasil penelitian dan implementasi sistem *back-end* MentalWell 1.0 menghasilkan kesimpulan sebagai berikut.

1. Pengembangan *back-end* menggunakan *framework* Hapi.js dilakukan melalui empat iterasi. Setiap iterasi disertai dengan pengujian *endpoint* menggunakan Postman untuk memastikan implementasi sesuai dengan kebutuhan sistem. Hasil pengujian menunjukkan bahwa seluruh fungsionalitas berjalan dengan output yang sesuai. *Framework* Hapi.js mendukung pengembangan sistem dengan struktur modular dan mampu memenuhi kebutuhan MentalWell 1.0. Hasil pengembangan ini telah berhasil diintegrasikan dengan sistem *front-end* melalui *endpoint* yang terdokumentasi, sehingga seluruh fungsi utama aplikasi dapat berjalan secara terpadu dan konsisten.
2. Sistem autentikasi dan otorisasi berbasis JSON Web Token (JWT) berhasil diimplementasikan, memastikan akses pengguna terbatas sesuai peran (pasien, psikolog, dan administrator) untuk menjaga keamanan dan privasi.
3. Database relasional PostgreSQL yang dikelola menggunakan Supabase dirancang secara terstruktur untuk menyimpan berbagai entitas data yang mendukung fungsionalitas sistem secara menyeluruh, mencakup informasi pengguna, aktivitas layanan, dan data transaksi.

5.2 Saran

Adapun saran untuk penelitian selanjutnya adalah sebagai berikut.

1. Untuk meningkatkan stabilitas sistem, disarankan untuk mengatur lingkungan *staging* dan *production* terpisah di Railway. Dengan memanfaatkan fitur *environment* pada Railway dan *branching* di GitHub (misalnya, *branch* '*develop'* untuk *staging* dan '*main'* untuk *production*), pengujian fitur baru dapat dilakukan tanpa mengganggu layanan aktif.
2. *Scheduled task*, seperti fungsi *startAutoUpdateCounselings(),* saat ini dijalankan dalam *instance* yang sama dengan server REST API utama menggunakan *setInterval*. Meskipun pendekatan ini cukup untuk tahap awal, pendekatan ini memiliki beberapa kelemahan, seperti pembagian sumber daya dengan proses utama, pencampuran *log* eksekusi, dan ketergantungan pada *licecycle* server. Jika server utama dihentikan atau dimulai ulang, *scheduled task* juga akan terhenti. Sebagai solusi, fungsi ini dapat dipisahkan ke dalam layanan terpisah menggunakan Railway *Function Service*. Pendekatan ini memungkinkan pembuatan *service* mandiri yang tidak bergantung pada *lifecycle* server utama. Dengan demikian, fungsi *auto-update* dapat dijalankan dalam *container* terpisah dengan *log* eksekusi yang terisolasi, skalabilitas, dan kemudahan pemantauan serta *debugging*.
3. Proses pembayaran saat ini masih bersifat manual melalui unggah bukti transfer. Untuk meningkatkan efisiensi dan memberikan pengalaman pengguna yang lebih baik, disarankan untuk mengintegrasikan sistem dengan *payment gateway*. Sebagai langkah awal, dapat digunakan layanan *sandbox* dari penyedia seperti Midtrans, DOKU, atau iPaymu untuk menguji alur pembayaran otomatis sebelum diterapkan pada lingkungan produksi.
4. Disarankan menambahkan implementasi CAPTCHA sebagai langkah meningkatkan keamanan sistem dari serangan bot otomatis. Fitur ini sebaiknya diterapkan pada halaman registrasi dan lupa kata sandi untuk memverifikasi bahwa tindakan dilakukan oleh manusia, sehingga dapat mengurangi risiko *brute-force* maupun pembuatan akun spam.

DAFTAR PUSTAKA

[1] C. M. Green, C. A. Hostutler, K. L. Lovero, J. A. Mautone, and R. Platt, "*Editorial: Pediatric integrated care: from vision to practice*," *Front. Psychiatry*, vol. 15, June 2024.

[2] N. Lisnarini, J. R. Suminar, and Y. Setianti, "Keunggulan dan Hambatan Komunikasi dalam Layanan Kesehatan Mental pada Aplikasi *Telemedicine* Halodoc," *PsikobuletinBuletin Ilm. Psikol.*, vol. 4, no. 3, Sept. 2023.

[3] I. Kurniawan and F. Rozi, "REST API Menggunakan NodeJS pada Aplikasi Transaksi Jasa Elektronik Berbasis Android," vol. 1, no. 4, 2020.

[4] M. F. Albaihaqi, "Rancang Bangun RESTful API dan Website Admin untuk Aplikasi Precision Agriculture," 2022.

[5] F. Doglio, *REST API Development with Node.js: Manage and Understand the Full Capabilities of Successful REST Development*. Apress, 2018.

[6] R. Kumar, *Ultimate Node. js for Cross-Platform App Development: Learn to Build Robust, Scalable, and Performant Server-Side JavaScript Applications with Node. js (English Edition)*, 1st ed. Delhi: Orange Education PVT Ltd, 2024.

[7] E. Bedor Hiland, *Therapy tech: the digital transformation of mental healthcare*. Minneapolis: University of Minnesota Press, 2021.

[8] M. Harrison, *Hapi.js in Action*. Manning Publications Co, 2017.

[9] K. Sud, *Practical Hapi: Build Your Own Hapi Apps and Learn from Industry Case Studies*. Berkeley, CA: Apress L. P, 2020.

[10] C. J. Ihrig, *Full Stack JavaScript Development with MEAN: MongoDB, Express, AngularJS, and Node. JS*, 1st ed. Victoria: SitePoint Pty, Limited, 2015.

[11] M. Fahreza, "Penerapan REST API Menggunakan JSON Web Token," *Fak. Sains Teknol. UIN Syarif Hidayatullah Jkt.*, 2023.

[12] "About," PostgreSQL. Accessed: Mar. 20, 2025. [Online]. Available: https://www.postgresql.org/about/

[13] "Realtime," Supabase. Accessed: Mar. 18, 2025. [Online]. Available: https://supabase.com/realtime

[14] “Storage Quickstart,” Supabase Docs. Accessed: Mar. 18, 2025. [Online]. Available: https://supabase.com/docs/guides/realtime

[15] Railway, “About Railway,” Railway Documentation. Accessed: Mar. 17, 2025. [Online]. Available: https://docs.railway.app/overview/about-railway

[16] “Nodemailer Documentation,” Nodemailer. Accessed: Mar. 20, 2025. [Online]. Available: https://nodemailer.com

[17] Luthfi, “Monitoring dan Visiualisasi Laporan SMPT TLS dengan Elastic Stack,” 2023.

[18] “Getting Started,” Axios. Accessed: Mar. 21, 2025. [Online]. Available: https://axios-http.com/docs/intro

[19] N. M. Syaifullah, “Rancangan Sistem Informasi Penjualan Berbasis Web Pada Home Industry Bubuk Kopi Ginjar Emas”.

[20] N. Salsabila, “Implementasi Back-End Pada Digitalisasi Pemesanan Menu Makanan dan Minuman Berbasis Website (Studi Kasus: Toko Mansure),” 2023.

[21] C.-H. Kung, *Software engineering*, Second edition. New York, NY: McGraw Hill LLC, 2024.

[22] R. Rahman, Niswa Ayu Lestari, and H. Hastuti, “Implementasi Metode Agile dan *Framework* Laravel Pada Pengembangan Sistem Informasi Bimbingan Skripsi Berbasis Web,” *Tek. Teknol. Inf. Dan Multimed.*, vol. 5, no. 2, Dec. 2024.

[23] I. Nawali and B. Renaldy Suteja, “Pembuatan Sistem Aplikasi Berbasis Website Konsultasi Orang Tua dengan Psikolog untuk Kesehatan Mental Anak,” *J. Strategi*, vol. 5, Mei 2023.

[24] L. X. Wen, “Counselling Services Management and Social Application,” *Inf. Syst. Eng.*, June 2022.

[25] S. Ahmed, M. Monirujjaman Khan, R. Alroobaea, and M. Masud, “Development of a Multi-feature Web-based Physiotherapy Service System,” *Intell. Autom. Soft Comput.*, vol. 29, no. 1, 2021.

[26] U. Zahroh, “*Back-End Design and Development on Rekaruang Application with Microservices Architecture*,” *JATISI J. Tek. Inform. Dan Sist. Inf.*, vol. 9, no. 1, pp. 86–96, Mar. 2022.

[27] V. Nejković, S. Cvetković, L. Stojadinović, and Đ. Đorđević, "*Design and Implementation of a Web-based Energy Data Visualization Platform Using Supabase and NextJS Frameworks,*" in *Proceedings SAUM 2024*, University of Niš, Faculty of Electronic Engineering, Faculty of Mechanical Engineering, Niš, 2024, pp. 67–70.

[28] S. Wang and H. Wang, "Digital Transformation in Real Estate Services: Development and Implementation of the Housing Selection Platform," *Front. Comput. Intell. Syst.*, vol. 8, no. 1, pp. 13–18, May 2024.

[29] "Brevo – *Platform to manage your customer relationships via email, SMS, WhatsApp, chat, and more*," Brevo. Accessed: Nov. 08, 2025. [Online]. Available: https://www.brevo.com/?r=t