

**LAPORAN MANAJEMEN PROYEK INFORMATIKA**  
**PEMBUATAN APLIKASI WEBSITE MEDICAL CONVERSATION**



Indra Dwi Aryadi	211011450468
Muhammad Rizki Ramadhan	211011450503
Lingga Safitri	211011450395
Prayoga Pratama	211011450555

**PROGRAM STUDI TEKNIK INFORMATIKA FAKULTAS ILMU**  
**KOMPUTER UNIVERSITAS PAMULANG**

**2025/2026**

# DAFTAR ISI

<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	1
1.3 Tujuan Penelitian.....	1
1.4 Manfaat Proyek/Penelitian .....	2
1.5 Ruang Lingkup Sistem .....	2
1.6. Metodologi Pengembangan Sistem .....	2
1.7. Sistematika Penulisan.....	3
<b>BAB II ANALISIS SISTEM .....</b>	<b>4</b>
2.1. Gambaran Umum Sistem / Instansi / Organisasi.....	4
2.2. Identifikasi Permasalahan Sistem Lama.....	4
2.3. Analisis Kebutuhan Sistem.....	5
2.3.1. Kebutuhan Fungsional .....	5
2.4. Spesifikasi Fungsional dan Non-Fungsional Sistem .....	6
2.4.1. Spesifikasi Fungsional .....	6
2.4.2. Spesifikasi Non-Fungsional .....	7
2.5. Use Case Diagram dan Penjelasan .....	8
2.5.1 Activity Diagram dan penjelasannya .....	9
2.5.2 Sequence diagram dan penjelasannya .....	16
2.5.3 class diagram dan penjelasannya .....	24
<b>BAB III IMPLEMENTASI SISTEM.....</b>	<b>26</b>
3.1. Arsitektur Sistem .....	26
3.2. Spesifikasi Perangkat Keras dan Lunak: .....	26
3.3. Front-end .....	27
A. UI/UX .....	27
3.4. Back-end.....	29
A. CDM (Conceptual Data Model) .....	29
B. LDM (Logical Data Model) .....	30
C. PDM (Physical Data Model) .....	30
D. SQL (Structured Query Language).....	31
3.5 Cuplikan Kode.....	32

1. Kode Backend: Endpoint untuk Membuat Pemesanan (File: index.js) .....	32
2. Kode Bot: Penanganan Pesan Masuk (File: bot.js) .....	33
3. Backend: Proses Login Pengguna dan Admin (index.js) .....	33
4. Bot: Logika Penerusan Pesan antara Pasien dan Dokter (bot.js) .....	34
3.6 Integrasi Modul Sistem .....	35
3.7 Panduan Instalasi dan Konfigurasi .....	36
<b>BAB IV PENGUJIAN DAN EVALUASI SISTEM .....</b>	<b>37</b>
4.1 Jenis Pengujian .....	37
4.1.1 Whitebox Testing .....	37
4.1.2 Blackbox Testing .....	51
4.2 Hasil Pengujian dan Analisis .....	55
4.2.1 Pengujian White-box dan Analisis .....	55
4.2.2 Pengujian Black-box dan Analisis .....	58
4.3 Evaluasi Sistem oleh Pengguna .....	60
4.4 Perbaikan Hasil Uji .....	63
1. Jika user atau pengguna lupa kata sandi itu bagaimana? .....	64
2. Feedback atau umpan balik dari dokter tidak terlihat di user? .....	69
3. Kenapa menggunakan admin.html? .....	70
4. Kenapa saat memasukkan form dokter/admin muncul error "forbidden"? .....	70
5. Darimana kita mendapatkan feedback setelah mengisi keluhan? .....	73
Tambahan: .....	73
<b>BAB V PENUTUP .....</b>	<b>81</b>
5.1 Kesimpulan .....	81
5.2 Saran Dan Pengembangan Lanjutan .....	81
5.3 Keterbatasan Sistem .....	83
<b>LAMPIRAN .....</b>	<b>84</b>

## DAFTAR GAMBAR

Gambar 2. 1 Use Case.....	8
Gambar 2.2 Activity Pendaftaran.....	9
Gambar 2. 3 Activity Login .....	11
Gambar 2.4 Activity Pemesan Makanan .....	12
Gambar 2.5 Activity Kelola Dokter .....	13
Gambar 2. 6 Activity Whatsapp.....	14
Gambar 2.7 Activity kode OTP .....	15
Gambar 2. 8 User dan Admin Login.....	16
Gambar 2.9 Kelola Dokter .....	18
Gambar 2.10 Pemesanan Konsultasi .....	19
Gambar 2. 11Konsultasi Melalui WhatsApp.....	20
Gambar 2.12 Kode OTP .....	22
Gambar 2.13class diagram dan penjelasannya .....	24
Gambar 4.14 Dashboard User atau pasien.....	43
Gambar 4.15 Index.js.....	71
Gambar 4.16.....	71
Gambar 4. 17 .....	72
Gambar 4.18 Admin.html .....	72
Gambar 4.19 .....	72
Gambar 4. 20 .....	73
Gambar 4.21 .....	74
Gambar 4.22 .....	74
Gambar 4.23 .....	74
Gambar 4.24 .....	75
Gambar 4.25 .....	75
Gambar 4.26 .....	76
Gambar 4.27 .....	77
Gambar 4.28 .....	78
Gambar 29 C:\User\MPI\Downloads\medical-convesation .....	88

## DAFTAR TABEL

Tabel 4 1 Login .....	37
Tabel 4 2 baris yang dilewati.....	38
Tabel 4 3 Baris Kode.....	39
Tabel 4 4 Dashboard User atau pasien .....	42
Tabel 4 5Baris Yang Dilewati .....	42
Tabel 4 6 Dashboard Admin.....	43
Tabel 4 7Baris Yang Dilewai .....	46
Tabel 4 8 OTP (One Time Password).....	47
Tabel 4 9 Baris yang dilewati .....	48
Tabel 4 10 Login .....	51
<i>Tabel 4 11 Register.....</i>	<i>51</i>
Tabel 4 12 Dashboard User atau pasien .....	52
Tabel 4 13 Dashboard Admin.....	53
Tabel 4 14 OTP (One Time Password).....	54
Tabel 4 15 Pertanyaan .....	61
Tabel 4 16 Perbaikan Hasil Uji.....	63

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Perkembangan teknologi informasi telah membawa perubahan signifikan dalam berbagai sektor, termasuk sektor kesehatan. Kebutuhan akan layanan kesehatan yang cepat, praktis, dan mudah diakses mendorong lahirnya inovasi layanan digital. Salah satu kebutuhan mendesak adalah sarana komunikasi yang efektif antara pasien dan tenaga medis, terutama untuk konsultasi awal atau gejala ringan.

Keterbatasan waktu, jarak, dan kondisi kesehatan seringkali menjadi kendala bagi pasien untuk datang langsung ke fasilitas kesehatan. Sementara itu, tenaga medis juga memerlukan media untuk menyaring kasus dan memberikan arahan awal tanpa interaksi fisik. Oleh karena itu, dibutuhkan sebuah aplikasi *web-based* yang memungkinkan percakapan medis antara pasien dan tenaga medis, yang dapat diakses kapan saja dan di mana saja.

Aplikasi "Medical Conversation" dirancang untuk menjembatani kebutuhan tersebut. Melalui platform ini, pasien dapat berkonsultasi secara real-time dengan tenaga medis, menyampaikan gejala, mengunggah dokumen medis, dan mendapatkan saran atau rujukan awal. Sistem ini tidak dimaksudkan untuk menggantikan konsultasi langsung, namun sebagai media awal komunikasi yang dapat meningkatkan efisiensi pelayanan kesehatan.

### **1.2 Rumusan Masalah**

Berdasarkan latar belakang di atas, maka rumusan masalah dalam penelitian ini adalah:

- A. Bagaimana merancang aplikasi berbasis web yang dapat digunakan sebagai media percakapan antara pasien dan tenaga medis?
- B. Fitur-fitur apa saja yang diperlukan dalam aplikasi untuk mendukung proses komunikasi yang efektif dan aman?
- C. Bagaimana cara menjaga keamanan dan kerahasiaan data percakapan medis antara pasien dan dokter?

### **1.3 Tujuan Penelitian**

Adapun tujuan dari penelitian ini adalah:

- a. Membangun aplikasi website yang mendukung komunikasi antara pasien dan tenaga medis secara real-time.

- b. Merancang fitur-fitur penting seperti live chat, unggah dokumen medis, dan manajemen riwayat konsultasi.
- c. Menerapkan sistem keamanan data yang menjamin privasi dan kerahasiaan informasi pengguna.

#### 1.4 Manfaat Proyek/Penelitian

Penelitian ini memberikan beberapa manfaat, antara lain:

- a. Bagi Pasien: Mempermudah akses awal ke layanan medis secara daring.
- b. Bagi Tenaga Medis: Membantu dalam melakukan penyaringan pasien berdasarkan gejala awal.
- c. Bagi Peneliti: Menjadi referensi dalam pengembangan aplikasi berbasis kesehatan.
- d. Bagi Masyarakat Umum: Meningkatkan kesadaran terhadap layanan kesehatan digital.

#### 1.5 Ruang Lingkup Sistem

Penelitian ini hanya mencakup pengembangan aplikasi website *Medical Conversation* untuk keperluan komunikasi awal antara pasien dan tenaga medis, tanpa melibatkan proses diagnosa atau resep resmi. Aplikasi dibangun menggunakan teknologi web dengan fitur dasar komunikasi, manajemen pengguna, dan pengamanan data.

#### 1.6. Metodologi Pengembangan Sistem

Pengembangan sistem "*Medical Conversation*" ini menggunakan pendekatan *agile* dengan model *prototyping*. Metodologi ini dipilih karena memungkinkan fleksibilitas dalam menghadapi perubahan kebutuhan selama proses pengembangan. Tahapan pengembangan meliputi:

1. **Perencanaan (Planning):** Mengidentifikasi tujuan utama proyek, yaitu membangun platform komunikasi antara pasien dan dokter, serta menentukan ruang lingkup dan fitur-fitur dasar.
2. **Analisis (Analysis):** Menganalisis kebutuhan pengguna (pasien dan admin) serta kebutuhan fungsional dan non-fungsional sistem. Pada tahap ini, dibuatlah *use case diagram* dan alur aktivitas untuk memvisualisasikan interaksi pengguna dengan sistem.
3. **Desain (Design):** Merancang arsitektur sistem, antarmuka pengguna (UI/UX), dan struktur basis data (CDM, LDM, PDM). Desain antarmuka difokuskan pada kemudahan penggunaan bagi pasien dan admin.
4. **Implementasi (Implementation):** Proses penulisan kode dilakukan dengan membagi tugas menjadi pengembangan *frontend*, *backend*, dan integrasi bot WhatsApp. *Backend* dibangun menggunakan Node.js dan Express.js, sementara *database* menggunakan SQLite.

5. **Pengujian (Testing):** Melakukan pengujian *White-box* untuk memvalidasi alur logika kode dan pengujian *Black-box* untuk memastikan semua fitur berjalan sesuai harapan dari perspektif pengguna.
6. **Evaluasi dan Iterasi:** Berdasarkan hasil pengujian dan umpan balik, dilakukan perbaikan dan penyesuaian secara berulang hingga sistem dianggap stabil dan memenuhi kebutuhan.

### 1.7. Sistematika Penulisan

- A. **BAB I: PENDAHULUAN:** Menjelaskan latar belakang, rumusan masalah, tujuan, manfaat, ruang lingkup, metodologi pengembangan, dan sistematika penulisan proyek.
- B. **BAB II: ANALISIS SISTEM:** Menguraikan gambaran umum sistem, analisis kebutuhan fungsional dan non-fungsional, serta pemodelan sistem menggunakan *Use Case Diagram*.
- C. **BAB III: IMPLEMENTASI SISTEM:** Merinci arsitektur sistem, spesifikasi perangkat keras dan lunak yang digunakan, hasil pengembangan antarmuka, basis data, cuplikan kode, integrasi modul, serta panduan instalasi.
- D. **BAB IV: PENGUJIAN DAN EVALUASI SISTEM:** Memaparkan jenis-jenis pengujian yang dilakukan (*Black-box* dan *White-box*), hasil analisis pengujian, evaluasi dari pengguna, serta perbaikan yang dilakukan berdasarkan hasil uji.
- E. **BAB V: PENUTUP:** Berisi kesimpulan dari keseluruhan proyek, saran untuk pengembangan lebih lanjut, dan keterbatasan sistem yang ada.



## BAB II

### ANALISIS SISTEM

#### 2.1. Gambaran Umum Sistem / Instansi / Organisasi

Sistem yang dikembangkan, "Medical Conversation," adalah sebuah aplikasi berbasis web yang berfungsi sebagai platform komunikasi digital antara pasien dan dokter. Proyek ini merupakan sistem mandiri yang dirancang dari awal untuk menjawab tantangan dalam layanan kesehatan modern. Aplikasi ini tidak terikat pada instansi atau organisasi kesehatan tertentu, sehingga memberikan fleksibilitas untuk dapat diimplementasikan di berbagai skala, mulai dari praktik dokter mandiri, klinik kecil, hingga institusi kesehatan yang lebih besar.

Tujuan utama dari sistem ini adalah untuk menjembatani kesenjangan komunikasi antara pasien dan tenaga medis dengan menyediakan sarana konsultasi awal yang efisien, aman, dan mudah diakses. Sistem ini memungkinkan pasien untuk melakukan pendaftaran, memilih dokter, dan mengirimkan keluhan medis secara daring. Selanjutnya, seluruh proses konsultasi dilakukan secara *real-time* melalui integrasi dengan bot WhatsApp, yang memastikan percakapan dapat berlangsung secara pribadi dan langsung antara kedua belah pihak.

#### 2.2. Identifikasi Permasalahan Sistem Lama

Proyek ini dikembangkan karena tidak adanya sistem spesifik sebelumnya untuk mengatasi masalah komunikasi medis secara daring. Permasalahan yang ada berasal dari metode konvensional, di antaranya:

1. **Keterbatasan Akses dan Waktu:** Pasien sering kali menghadapi kendala waktu dan jarak untuk dapat berkonsultasi langsung dengan dokter, terutama untuk keluhan yang bersifat ringan atau memerlukan tindak lanjut sederhana.
2. **Inefisiensi Komunikasi:** Komunikasi melalui telepon atau pesan singkat konvensional tidak terstruktur, tidak tercatat dengan baik, dan sulit untuk dilampirkan dokumen pendukung medis.
3. **Beban Tenaga Medis:** Tenaga medis memerlukan sebuah media untuk dapat melakukan penyaringan awal terhadap keluhan pasien, sehingga dapat memprioritaskan kasus yang lebih mendesak dan memberikan arahan awal tanpa harus melakukan tatap muka.
4. **Kerahasiaan Data:** Komunikasi melalui platform publik rentan terhadap isu privasi dan keamanan data medis pasien yang sensitif.

Aplikasi "Medical Conversation" dirancang untuk menjadi solusi atas permasalahan-permasalahan tersebut dengan menyediakan platform yang terpusat, aman, dan terstruktur.

## 2.3. Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dibagi menjadi dua kategori utama, yaitu kebutuhan fungsional yang mendefinisikan fitur-fitur sistem, dan kebutuhan non-fungsional yang mendefinisikan kualitas dan batasan dari sistem.

### 2.3.1. Kebutuhan Fungsional

Kebutuhan fungsional sistem ini adalah sebagai berikut:

#### 1. Manajemen Pengguna:

- Sistem harus dapat membedakan tiga peran pengguna: Admin, Pasien (User), dan Dokter.
- Sistem harus menyediakan fitur

**Pendaftaran** bagi pengguna baru (pasien) dengan validasi input seperti format nomor telepon dan kekuatan kata sandi.

- Sistem harus menyediakan fitur

**Login** untuk semua peran (Admin dan Pasien) dengan mekanisme autentikasi yang aman.

- Sistem harus menyediakan fitur

**Lupa Password** menggunakan verifikasi kode OTP yang dikirim melalui WhatsApp.

- Admin harus dapat melihat daftar semua pengguna dan menghapus akun pengguna.

#### 2. Manajemen Dokter:

- Admin harus dapat menambah, melihat, dan menghapus data dokter.
- Data dokter yang dikelola mencakup nama, spesialisasi, nomor WhatsApp, dan URL foto.

#### 3. Proses Konsultasi:

- Pasien dapat melihat daftar dokter yang tersedia setelah login.
- Pasien dapat memilih dokter dan mengirimkan keluhan awal melalui formulir di aplikasi web.
- Sistem harus dapat memvalidasi bahwa pasien tidak dapat memesan konsultasi baru jika masih memiliki sesi aktif dengan dokter lain.
- Sistem harus meneruskan keluhan pasien ke nomor WhatsApp dokter yang dipilih melalui bot.
- Sistem harus meneruskan balasan dari dokter kepada pasien melalui WhatsApp.

#### 4. **Fitur Bot WhatsApp:**

- Bot harus dapat mengirimkan menu perintah yang tersedia bagi pengguna.
- Bot harus menyediakan fitur interaksi dengan AI (Asisten Dokter) untuk menjawab pertanyaan medis umum.
- Bot harus dapat mengakhiri sesi konsultasi berdasarkan perintah dari pasien atau dokter.

### 2.3.2. Kebutuhan Non-Fungsional

#### 1. **Keamanan (Security):**

- Sistem harus melindungi dari serangan *Cross-Site Scripting* (XSS) dengan melakukan sanitasi input.
- Sistem harus menggunakan token CSRF (*Cross-Site Request Forgery*) untuk melindungi *endpoint* dari permintaan yang tidak sah.
- Sistem harus menerapkan pembatasan laju permintaan (*rate limiting*) pada fitur login, registrasi, dan OTP untuk mencegah serangan *brute-force*.
- Sesi pengguna harus dikelola dengan aman dan memiliki batas waktu.

#### 2. **Ketersediaan (Availability):** Aplikasi web dan bot WhatsApp harus dapat diakses oleh pengguna setiap saat (24/7) dengan *downtime* yang minimal.

#### 3. **Kemudahan Penggunaan (Usability):** Antarmuka pengguna harus dirancang secara intuitif dan mudah dipahami, baik untuk pasien maupun admin, sehingga tidak memerlukan pelatihan khusus.

#### 4. **Kinerja (Performance):** Sistem harus memberikan respons yang cepat, terutama pada proses pengiriman dan penerimaan pesan antara pasien dan dokter.

#### 5. **Perawatan (Maintainability):** Kode sumber harus ditulis dengan struktur yang baik dan terorganisir (misalnya, pemisahan antara logika *routing*, bot, dan utilitas) agar mudah untuk dipelihara dan dikembangkan di masa mendatang.

### 2.4. Spesifikasi Fungsional dan Non-Fungsional Sistem

Spesifikasi detail dari kebutuhan yang telah diidentifikasi adalah sebagai berikut:

#### 2.4.1. Spesifikasi Fungsional

- **Pendaftaran Pengguna:** Aktor: Pasien. Pasien mengisi nama, nomor telepon (sebagai *username*), dan kata sandi. Sistem memvalidasi input dan menyimpan data ke tabel users dengan peran 'patient'.
- **Login Pengguna:** Aktor: Pasien, Admin. Pengguna memasukkan *username* dan kata sandi. Sistem akan memeriksa kredensial terhadap data di tabel users atau kredensial admin yang disimpan di

*environment variable*. Jika berhasil, sistem akan membuat sesi dan mengarahkan pengguna ke *dashboard* yang sesuai.

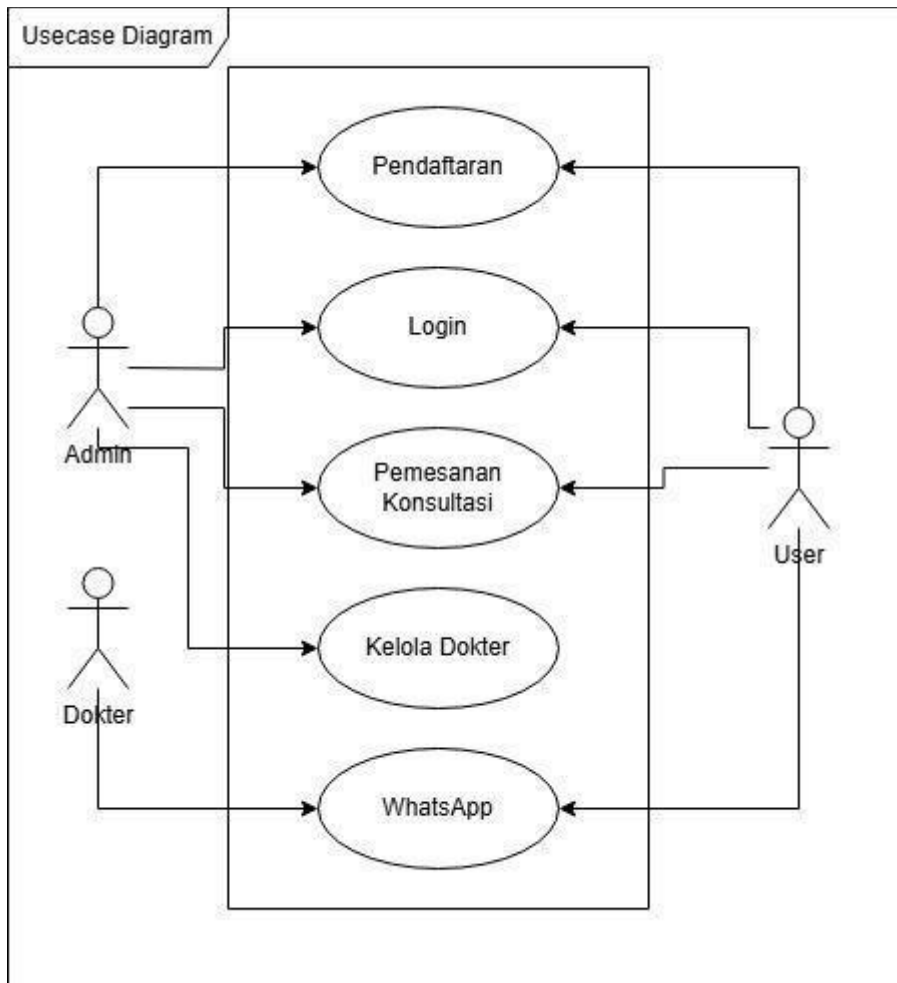
- **Pemesanan Konsultasi:** Aktor: Pasien. Pasien memilih dokter dari daftar, menulis keluhan, dan mengirimkannya. Sistem akan membuat entri baru di tabel bookings yang menautkan ID pasien, ID dokter, dan pesan keluhan.
- **Kelola Dokter:** Aktor: Admin. Admin mengakses halaman admin, mengisi formulir tambah dokter, dan mengirimkannya. Data akan disimpan di tabel doctors. Admin juga dapat menghapus data dokter dari daftar.
- **Interaksi WhatsApp:** Aktor: Pasien, Dokter. Bot mendengarkan pesan masuk. Jika pesan berasal dari dokter yang terkait dengan sesi aktif, pesan diteruskan ke pasien. Jika dari pasien, pesan diteruskan ke dokter. Percakapan disimpan di tabel messages.

#### 2.4.2. Spesifikasi Non-Fungsional

- **Keamanan Data:** Data sensitif seperti kata sandi tidak disimpan dalam bentuk *plain text*. Komunikasi antara *frontend* dan *backend* diamankan menggunakan *header* CSRF. Library **Helmet.js** digunakan untuk menerapkan *header* HTTP yang aman.
- **Antarmuka Responsif:** Desain antarmuka dibuat responsif sehingga dapat diakses dengan baik melalui perangkat *desktop* maupun *mobile browser*.
- **Skalabilitas:** Meskipun menggunakan SQLite yang berbasis file, arsitektur aplikasi memungkinkan penggantian sistem *database* ke sistem yang lebih skalabel seperti MySQL atau PostgreSQL di masa depan dengan perubahan minimal pada logika aplikasi.

## 2.5. Use Case Diagram dan Penjelasan

### Penjelasan Use Case:



Gambar 2. 1 Use Case

#### A. Pendaftaran:

- **Aktor:** User (Pasien)
- **Deskripsi:** Pengguna baru dapat mendaftarkan akun dengan memberikan nama, nomor telepon, dan kata sandi. Sistem akan memvalidasi data dan membuat akun baru. Admin dan Dokter tidak melakukan pendaftaran melalui sistem ini.

#### B. Login:

- **Aktor:** User (Pasien), Admin
- **Deskripsi:** Pengguna yang telah terdaftar (Pasien dan Admin) dapat masuk ke sistem menggunakan *username* dan kata sandi. Sistem akan mengarahkan mereka ke *dashboard* masing-masing sesuai dengan perannya.

#### C. Pemesanan Konsultasi:

- **Aktor:** User (Pasien)
- **Deskripsi:** Setelah login, pasien dapat melihat daftar dokter, memilih salah satu, dan menuliskan keluhan medis untuk memulai sesi konsultasi.

#### D. Kelola Dokter:

- **Aktor:** Admin
- **Deskripsi:** Admin memiliki hak akses untuk menambah data dokter baru, melihat daftar dokter yang ada, dan menghapus data dokter dari sistem.

#### E. Interaksi WhatsApp:

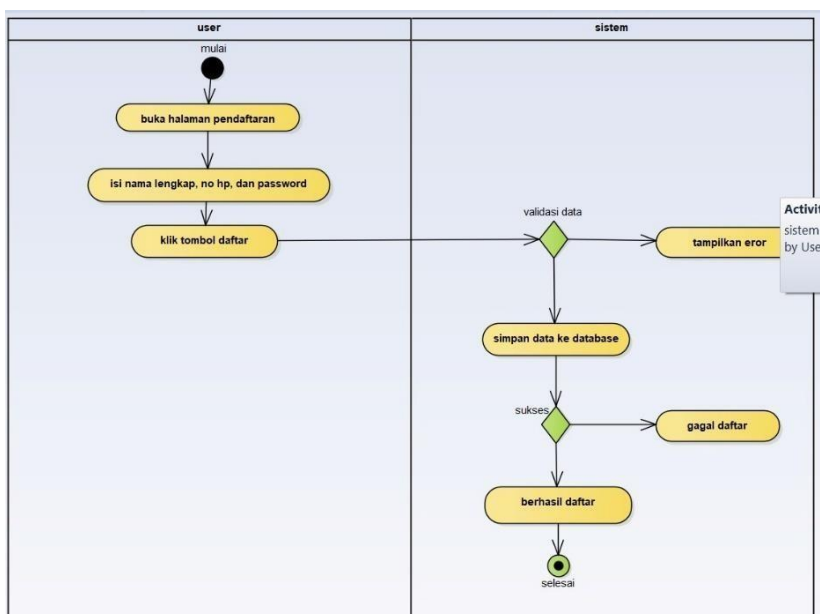
- **Aktor:** User (Pasien), Dokter
- **Deskripsi:** Ini adalah *use case* ini dimana sistem, melalui bot WhatsApp, memfasilitasi percakapan dua arah. Pasien mengirim pesan ke dokter, dan dokter membalas pesan tersebut. Seluruh interaksi ini dimediasi oleh sistem. Baik pasien maupun dokter juga dapat berinteraksi dengan fitur AI yang disediakan bot.

#### F. Kode OTP:

- **Aktor:** User (Pasien)
- **Deskripsi:** ini adalah use case yang Dimana system akan melakukan pergantian password berdasarkan user atau pasien yang sudah terdaftar, apabila user tersebut lupa password. Password tersebut menggunakan kode otp yang hanya akan valid selama 5 menit dan tidak lebih dari itu

### 2.5.1 Activity Diagram dan penjelasannya

#### A. Activity Pendaftaran



Gambar 2.2 Activity Pendaftaran

## Penjelasan Diagram

### 1. Mulai (Start)

- Tanda awal (bulat hitam) menunjukkan titik awal dari proses pendaftaran.

### 2. User Activities

- Buka Halaman Pendaftaran
  - Pengguna membuka halaman untuk mendaftar.
- Isi Nama Lengkap, No HP, dan Password
  - Pengguna menginput data yang diperlukan dalam formulir pendaftaran.
- Klik Tombol Daftar
  - Pengguna menekan tombol untuk mengirim data pendaftaran.

### 1. Sistem Activities

- Validasi Data
  - Sistem memeriksa keabsahan informasi yang diinput oleh pengguna.
  - Dua kemungkinan hasil:
    - Jika valid, lanjut ke langkah berikutnya.
    - Jika invalid, kembali ke pengguna dengan "Tampilkan Error" untuk meminta perbaikan.

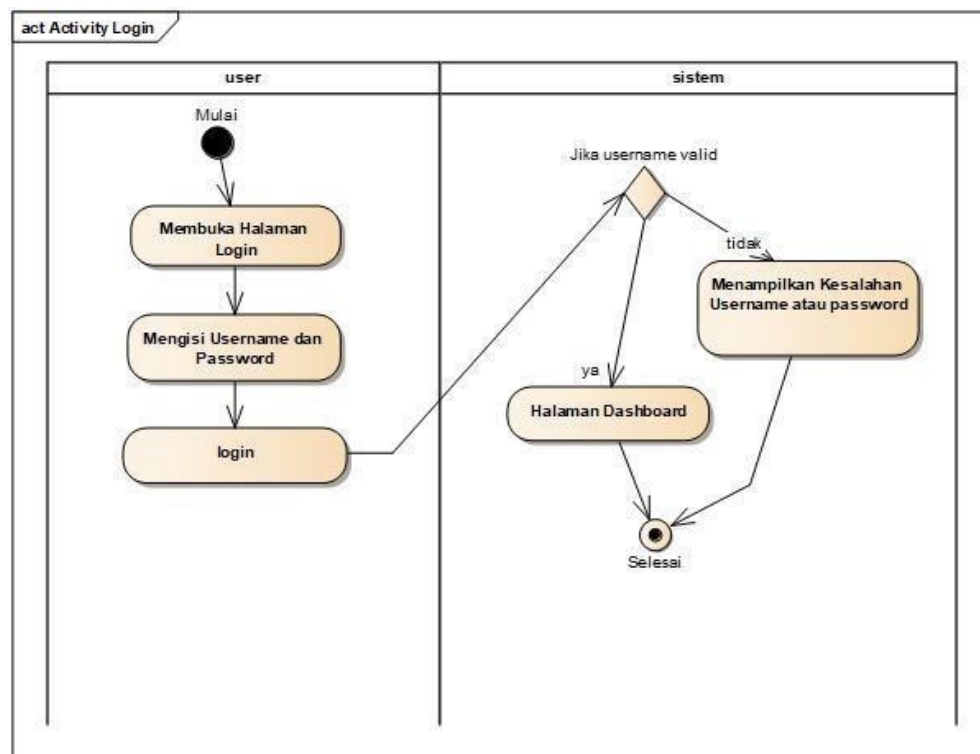
### 2. Penyimpanan Data

- Jika data valid, sistem akan:
  - Simpan Data ke Database
    - Data yang telah disetujui akan disimpan ke dalam basis data.
  - Dua kemungkinan hasil:
    - Berhasil Daftar: Data tersimpan dengan baik.
    - Gagal Daftar: Terjadi kesalahan saat menyimpan data.

### 3. Selesai (End)

- Proses pendaftaran berakhir dengan titik akhir (lingkaran hitam dengan pinggiran putih) setelah menginformasikan pengguna tentang status pendaftaran.

## B. Activity Login



Gambar 2. 3 Activity Login

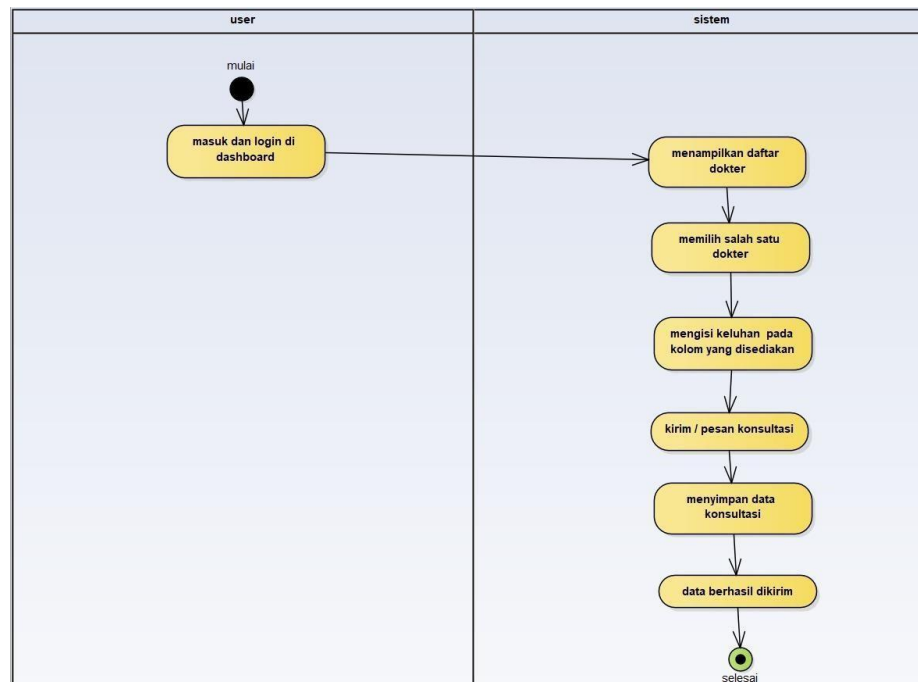
### Diagram Aktivitas Login

- **Pengguna (User)**
  - **Mulai:** Proses dimulai.
  - **Membuka Halaman Login:** Pengguna membuka halaman untuk login.
  - **Mengisi Username dan Password:** Pengguna memasukkan informasi login.
  - **Login:** Pengguna menekan tombol login.
- **Sistem**
  - **Jika Username Valid:** Sistem memeriksa validitas username.
    - **Ya:** Sistem mengarahkan ke:
      - **Halaman Dashboard:** Pengguna diarahkan ke dashboard setelah login berhasil.



- **Tidak:** Sistem menampilkan:
  - **Kesalahan Username atau Password:** Pesan kesalahan jika informasi login tidak valid.
- **Selesai:** Proses login selesai.

### C. Activity Pemesanan Konsultasi



Gambar 2.4 Activity Pemesanan Makanan

### Diagram Proses Pengguna dan Sistem

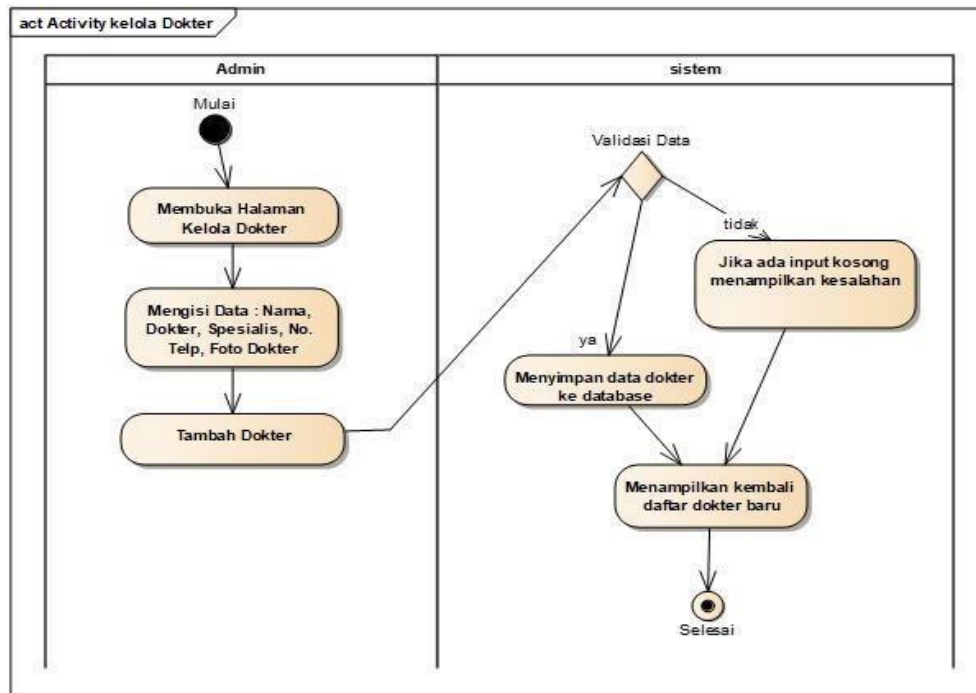
#### 1. Pengguna (User)

- Memulai
  - Masuk dan login di dashboard

#### 2. Sistem

- Menampilkan daftar dokter
  - Memilih salah satu dokter
  - Mengisi keluhan pada kolom yang disediakan
- Kirim pesan konsultasi
- Menyimpan data konsultasi
- Data berhasil dikirim

## D. Activity Kelola Dokter



Gambar 2.5 Activity Kelola Dokter

Berikut adalah ringkasan dari diagram aktivitas "Kelola Dokter" yang ditampilkan:

### 1. Awal Proses:

- **Admin** membuka halaman "Kelola Dokter".

### 2. Menggabungkan Data:

- Admin menginput data:
  - Nama,
  - Dokter,
  - Spesialis,
  - No. Telp,
  - Foto Dokter.

### 3. Tambah Dokter:

- Admin memilih opsi untuk menambah dokter.

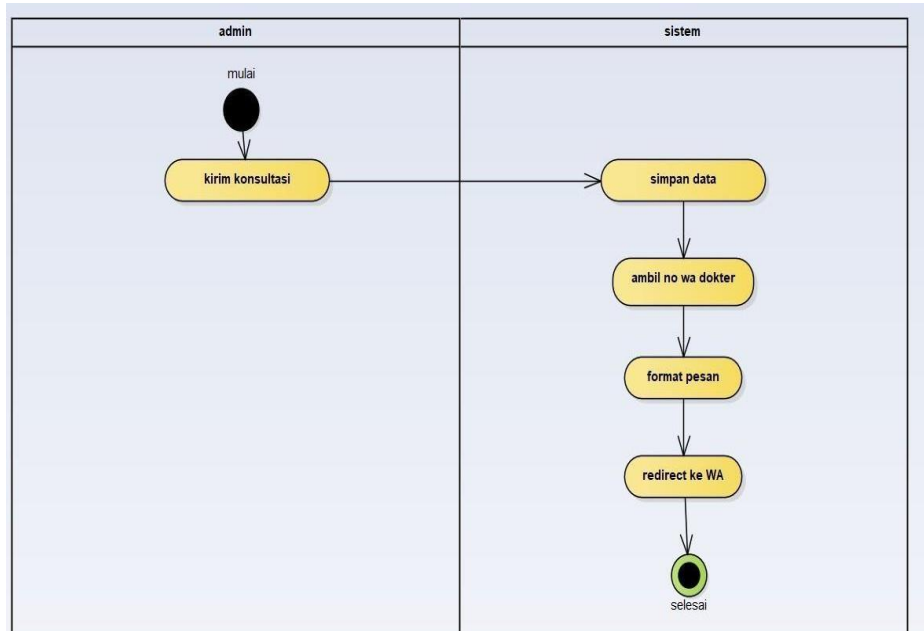
### 4. Validasi Data oleh Sistem:

- Sistem melakukan validasi data:
  - **Jika tidak ada input kosong:**
    - Menyimpan data dokter ke database.
  - **Jika ada input kosong:**
    - Menampilkan kesalahan dan mengarahkan kembali ke daftar dokter baru.

## 5. Akhir Proses:

- Proses selesai.

## E. Activity whatsapp



Gambar 2. 6 Activity Whatsapp

## Diagram Alur Pengiriman Konsultasi

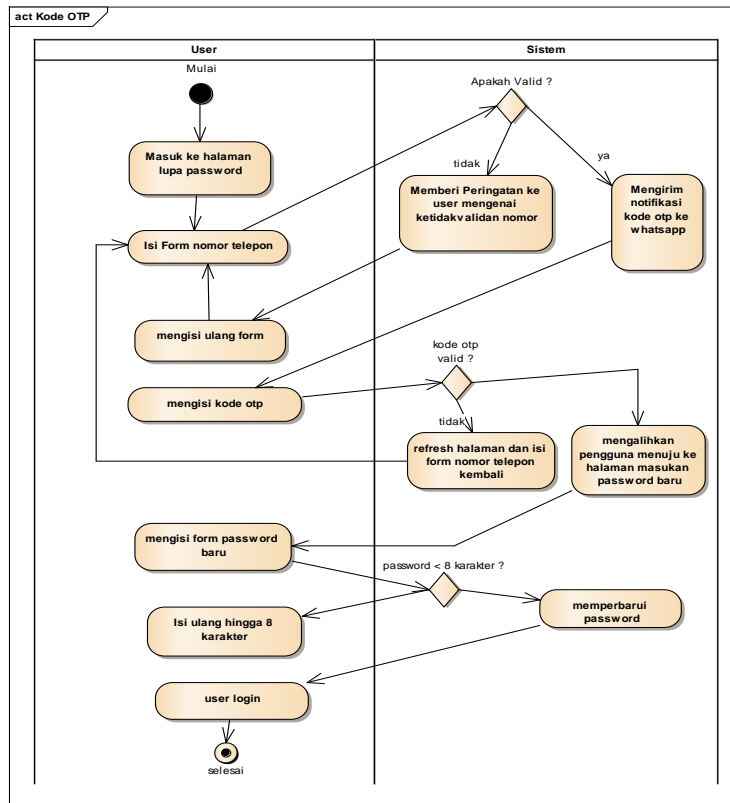
### 1. Admin

- **Mulai**
- **Kirim Konsultasi**
  - Langkah ini menandakan bahwa admin memulai proses dengan mengirimkan konsultasi.

### 2. Sistem

- **Simpan Data**
  - Sistem akan menyimpan data yang diterima dari admin.
- **Ambil Nomor WhatsApp Dokter**
  - Sistem mengambil nomor WhatsApp dari dokter yang relevan.
- **Format Pesan**
  - Sistem memformat pesan berdasarkan informasi yang telah disimpan.
- **Redirect ke WhatsApp**
  - Sistem mengalihkan pesan yang telah diformat ke aplikasi WhatsApp.
- **Selesai**
  - Proses pengiriman konsultasi selesai.

## F. Activity kode OTP



Gambar 2.7 Activity kode OTP

### Proses Penggunaan Kode OTP

#### 1. Masuk ke Halaman Lupa Password

- Isi form dengan email atau nomor telepon.
- Mengisi ulang form.
- Mengklik tombol kirim.

#### 2. Menunggu Penerimaan Kode

- Memastikan penerimaan kode di email atau nomor telepon.
- Mengingat dan mencatat kode yang diterima.

#### 3. Mengisi Kode di Form

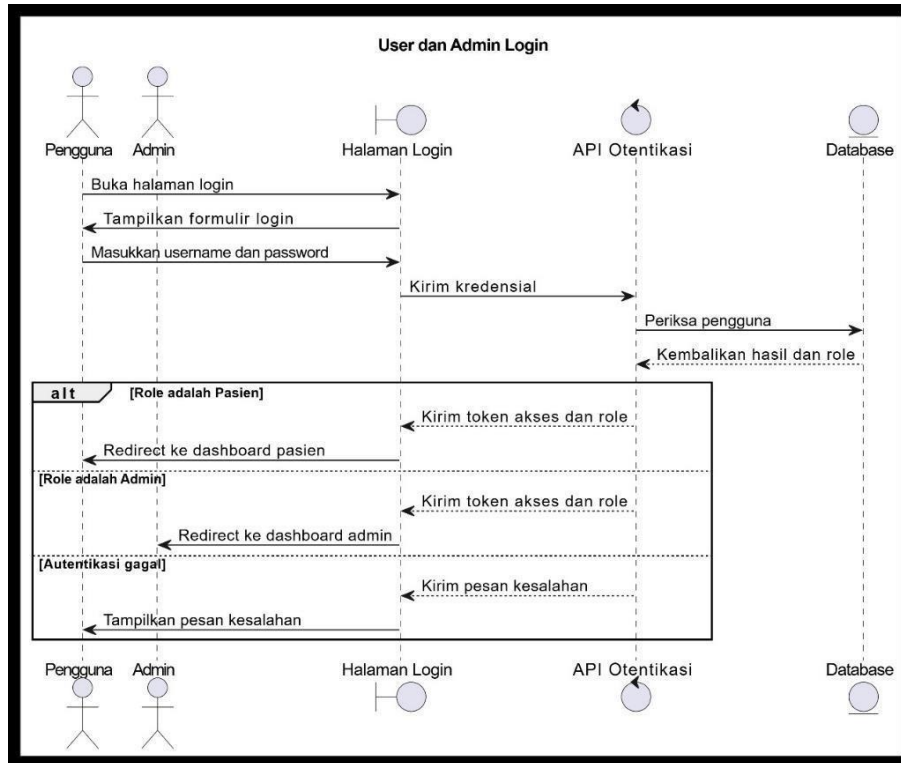
- Mengisi form dengan kode yang diterima.
- Mengonfirmasi untuk melanjutkan ke langkah berikutnya.

#### 4. Mengatur Password Baru

- Mengisi form password baru.
- Mengonfirmasi password baru.
- Menyimpan perubahan password.
- Melakukan login dengan password baru.

## 2.5.2 Sequence diagram dan penjelasannya

### A. User dan Admin Login



Gambar 2. 8 User dan Admin Login

### Penjelasan Sequence Diagram

#### 1. Pengguna dan Admin

- Terdapat dua aktor dalam diagram ini: Pengguna (User) dan Admin.

#### 2. Proses Login

- **Buka Halaman Login:** Pengguna atau Admin membuka halaman login.
- **Tampilkan Formulir Login:** Sistem menampilkan formulir yang meminta pengguna untuk memasukkan username dan password.
- **Masukkan Username dan Password:** Pengguna mengisi data login.

#### 3. Kirim Kredensial

- Setelah input, kredensial (username dan password) dikirim ke API Otentikasi untuk diperiksa.

#### 4. Pemeriksaan Pengguna

- **Periksa Kredensial:** API memeriksa apakah kredensial yang diberikan valid dengan melakukan query ke Database.
- **Kembalikan Hasil dan Role:** Jika valid, API mengembalikan hasil validasi dan role pengguna (apakah sebagai pasien atau admin).

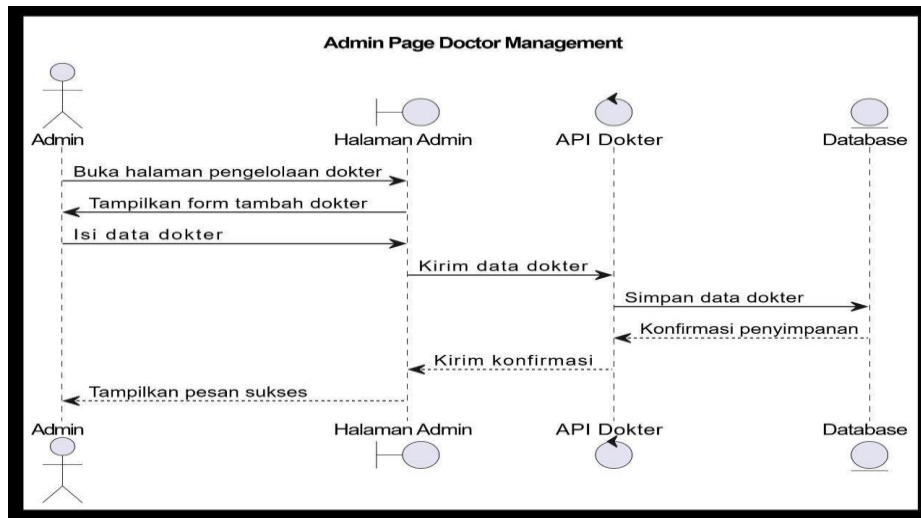
## 5. Perlakuan Berdasarkan Role

- **alt (Role adalah Pasien):**
  - Jika role adalah pasien:
    - **Kirim Token Akses dan Role:** Sistem mengirimkan token akses kepada pengguna dan mengarahkan ke dashboard pasien.
- **Role adalah Admin:**
  - Jika role adalah admin:
    - **Redirect ke Dashboard Admin:** Sistem mengarahkan admin ke dashboard admin dengan mengirimkan token akses yang diperlukan.

## 6. Autentikasi Gagal

- Jika kredensial tidak valid:
  - **Tampilkan Pesan Kesalahan:** Sistem akan menampilkan pesan kesalahan kepada pengguna, memberitahukan bahwa login gagal.

## B. Kelola Dokter



Gambar 2.9 Kelola Dokter

Berikut adalah penjelasan mengenai diagram manajemen halaman admin dokter yang telah diberikan:

### 1. Konteks Umum

- Diagram ini menggambarkan proses manajemen data dokter dalam sebuah sistem dengan peran admin yang berinteraksi dengan antarmuka dan basis data.

### 2. Aktor dan Entitas

- **Admin:** Pengguna yang mengelola data dokter.
- **Halaman Admin:** Antarmuka tempat admin berinteraksi.
- **API Dokter:** Proses backend yang menangani permintaan terkait data dokter.
- **Database:** Penyimpanan permanen untuk data dokter.

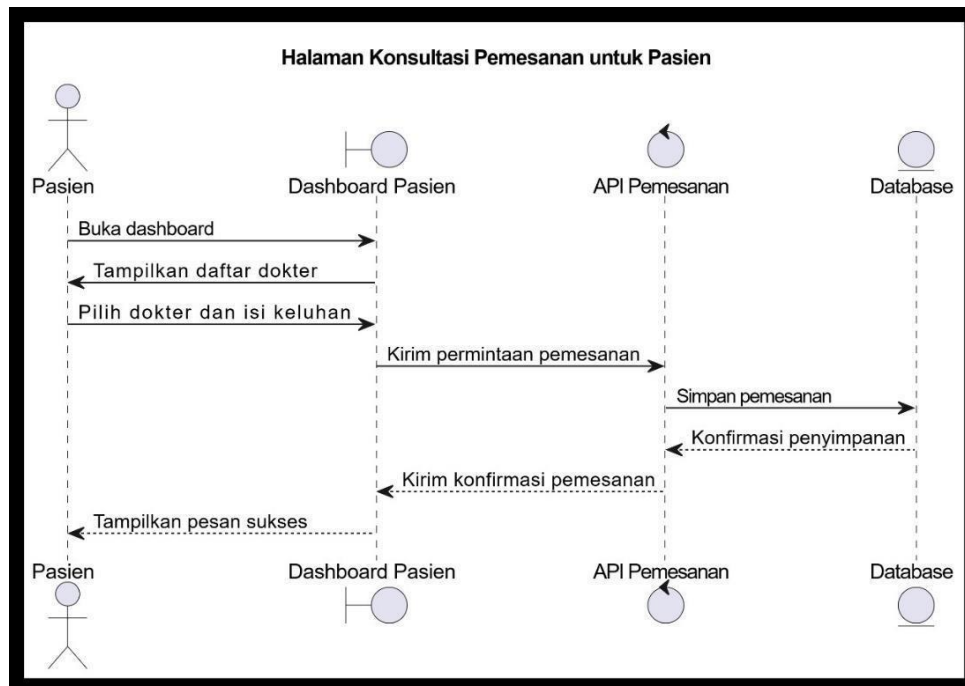
### 3. Proses Utama

- **Buka Halaman Pengelolaan Dokter:** Admin memulai dengan membuka antarmuka pengelolaan dokter.
- **Tampilkan Form Tambah Dokter:** Sistem menampilkan formulir untuk menambahkan data dokter.
- **Isi Data Dokter:** Admin mengisi informasi yang diperlukan dalam formulir.
- **Kirim Data Dokter:** Data yang diisi admin dikirim ke API Dokter.
- **Simpan Data Dokter:** API menyimpan data yang diterima ke dalam database.
- **Kirim Konfirmasi:** Setelah penyimpanan berhasil, API mengirimkan konfirmasi kepada halaman admin.
- **Tampilkan Pesan Sukses:** Halaman admin menampilkan pesan sukses yang mengonfirmasi bahwa data dokter berhasil disimpan.

### 4. Alur Informasi

- Terdapat alur antara admin, halaman admin, API, dan database yang menunjukkan interaksi dan pertukaran data yang terjadi selama proses.

### C. Pemesanan Konsultasi



Gambar 2.10 Pemesanan Konsultasi

Diagram ini menggambarkan alur proses pemesanan konsultasi bagi pasien. Berikut adalah langkah-langkah yang tercantum dalam diagram tersebut:

#### 1. Pembukaan Dashboard

- Pasien membuka dashboard pemesanan.
- Sistem menampilkan daftar dokter yang tersedia.

#### 2. Pemilihan Dokter

- Pasien memilih dokter yang diinginkan.
- Pasien mengisi keluhan yang ingin disampaikan.

#### 3. Pengiriman Permintaan Pemesanan

- Setelah memilih dokter dan mengisi keluhan, pasien mengirimkan permintaan pemesanan.
- Permintaan tersebut dikirim melalui API Pemesanan untuk diproses.

#### 4. Penyimpanan dan Konfirmasi



- Permintaan pemesanan disimpan di Database.
- Sistem mengirimkan konfirmasi penyimpanan ke API Pemesanan.

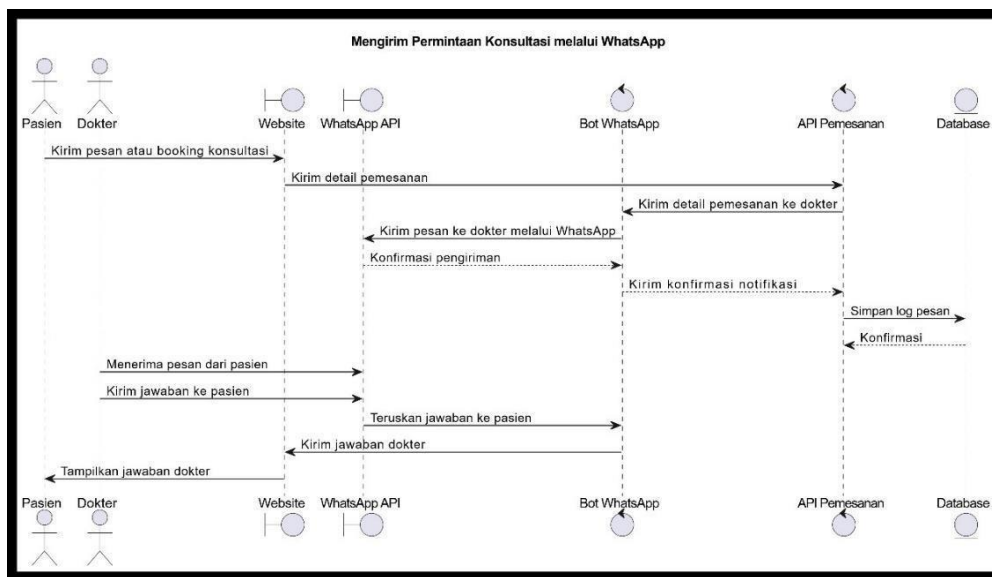
## 5. Menampilkan Pesan Sukses

- Setelah konfirmasi diterima, pasien akan dapat melihat pesan sukses di dashboard.

## 6. Pengulangan Proses

- Pasien dapat kembali ke dashboard untuk pemesanan selanjutnya.

## D. Konsultasi Melalui WhatsApp



Gambar 2. 11Konsultasi Melalui WhatsApp

Berikut adalah penjelasan mengenai diagram alur yang menggambarkan proses pengiriman permintaan konsultasi melalui WhatsApp:

### 1. Entitas Utama

- **Pasien:** Individu yang ingin melakukan konsultasi.
- **Dokter:** Profesional kesehatan yang memberikan konsultasi.
- **Website:** Platform yang digunakan untuk mengirimkan permintaan konsultasi.
- **WhatsApp API:** Antarmuka pemrograman aplikasi yang menghubungkan platform WhatsApp dengan sistem.
- **Bot WhatsApp:** Otomatisasi yang berfungsi untuk mengelola pesan.
- **Database:** Tempat penyimpanan data pasien, dokter, dan permintaan konsultasi.

## **2. Proses Perjalanan Pengguna**

- **Inisiasi Permintaan:**
  - Pasien mengirimkan pesan atau melakukan booking konsultasi melalui website atau WhatsApp.
- **Penerimaan Pesan:**
  - Website atau WhatsApp API menerima pesan dari pasien dan meneruskan detail permintaan ke Bot WhatsApp.
- **Pengiriman Detail:**
  - Bot WhatsApp mengirimkan detail permintaan konsultasi ke dokter melalui WhatsApp.
- **Konfirmasi:**
  - Dokter menerima pesan dan mengirimkan konfirmasi permintaan kembali melalui sistem.

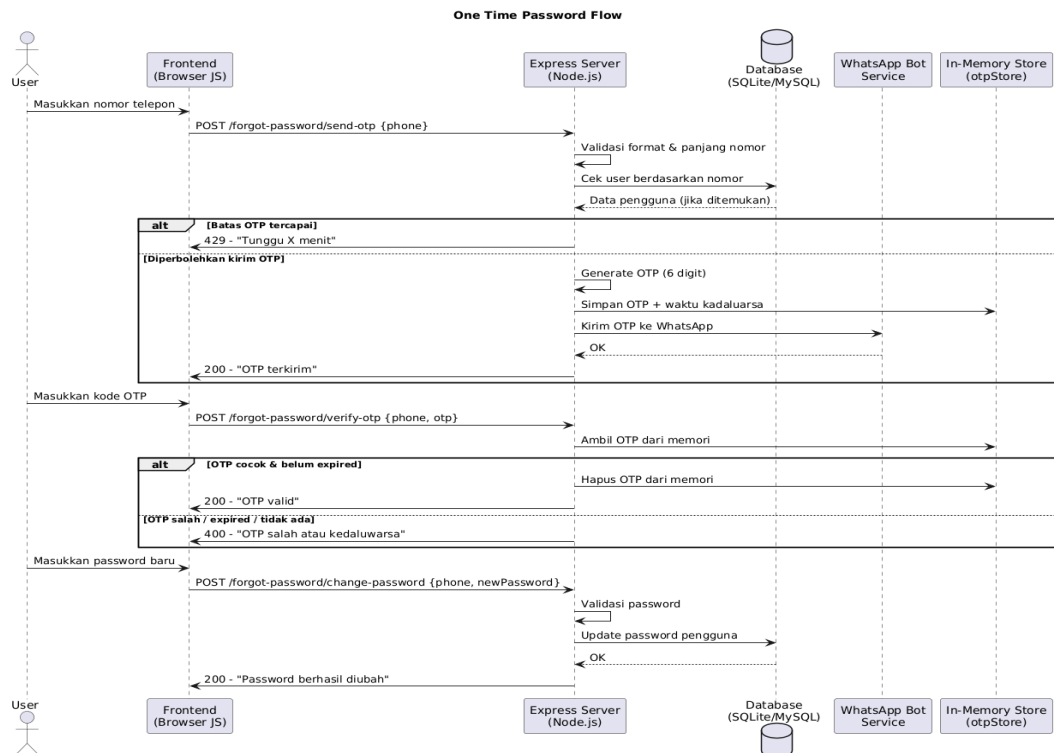
## **3. Tindakan Setelah Konfirmasi**

- **Tindak Lanjut:**
  - Bot WhatsApp menerima konfirmasi dari dokter dan menginformasikan pasien melalui pesan WhatsApp.
- **Jawaban dari Dokter:**
  - Dokter memberikan jawaban atau konsultasi terkait pertanyaan pasien dan mengirimkannya kembali ke sistem.
- **Pengiriman Jawaban:**
  - Jawaban dokter diteruskan oleh Bot WhatsApp kepada pasien.

## **4. Penyimpanan Data**

- Semua transaksi dan komunikasi dapat disimpan dalam database untuk keperluan dokumentasi dan analisis lebih lanjut.

## E. Kode OTP



Gambar 2.12 Kode OTP

Berikut adalah penjelasan mengenai **One Time Password Flow** yang ditunjukkan dalam diagram tersebut:

### 1. Proses Awal

- **User Masukkan Nomor Telepon**

- Pengguna memasukkan nomor telepon mereka di frontend (antarmuka pengguna).
- Sistem mengirimkan permintaan POST ke endpoint **/forgot-password/send-otp** dengan nomor telepon.

### 2. Validasi Nomor Telepon

- **Validasi Format dan Pencarian User**

- Server melakukan validasi format nomor telepon yang dimasukkan.
- Memeriksa data pengguna di database untuk memastikan nomor telepon tersebut terdaftar.

### 3. Pengiriman OTP

- **Menghasilkan OTP**

- Jika nomor telepon valid, sistem akan menghasilkan OTP (One Time Password) berupa 6 digit.
- OTP disimpan dengan batas waktu kedaluwarsa yang telah ditentukan.
- OTP kemudian dikirimkan ke pengguna melalui WhatsApp.
- **Respon Pengiriman**
  - Sistem mengirimkan respon dengan status 200 dan pesan "OTP terkirim."

#### 4. Memasukkan Kode OTP

- **User Memasukkan Kode OTP**
  - Pengguna diminta untuk memasukkan kode OTP yang diterima.
  - Pengguna mengirimkan permintaan POST ke endpoint **/forgot-password/verify-otp** dengan nomor telepon dan kode OTP.

#### 5. Validasi OTP

- **Memeriksa Kode OTP**
  - Server memeriksa apakah OTP yang dimasukkan valid dan belum kedaluwarsa.
  - Jika OTP valid, sistem akan menghapus OTP dari penyimpanan untuk mencegah penggunaan ulang.
- **Respon Validasi OTP**
  - Jika OTP valid, sistem mengirimkan respon dengan status 200 dan pesan "OTP valid." Jika tidak, pesan kesalahan dikirim.

#### 6. Mengubah Password

- **User Memasukkan Password Baru**
  - Setelah OTP terverifikasi, pengguna dapat memasukkan password baru.
  - Pengguna mengirimkan permintaan POST ke endpoint **/forgot-password/change-password** dengan password baru.

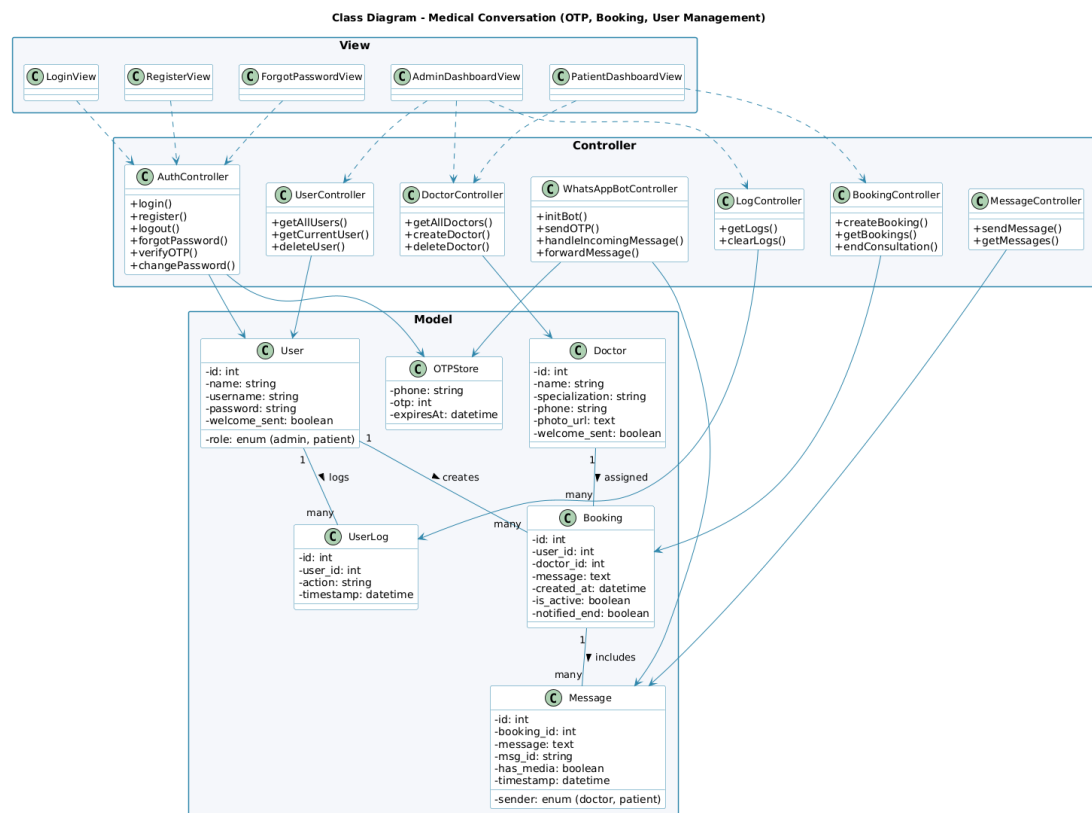
#### 7. Pembaruan Password

- **Validasi dan Pembaruan Password**
  - Server memvalidasi password baru dan melakukan pembaruan di basis data.

- **Respon Pembaruan**

- Sistem mengirimkan respon dengan status 200 dan pesan "Password berhasil diubah."

### 2.5.3 class diagram dan penjelasannya



Gambar 2.13 class diagram dan penjelasannya

#### A. Penjelasan Diagram Kelas - Medical Conversation (OTP, Booking, User Management)

Diagram ini menggambarkan struktur dari sistem yang berfungsi dalam mengelola percakapan medis, booking, dan manajemen pengguna. Berikut adalah detail elemen-elemen yang ada dalam diagram:

##### 1. View

- **LoginView:** Tampilan untuk pengguna masuk ke sistem.
- **RegisterView:** Tampilan untuk pengguna yang ingin mendaftar.
- **ForgotPasswordView:** Tampilan untuk memulihkan password yang hilang.
- **AdminDashboardView:** Antarmuka untuk admin mengelola sistem.

- **PatientDashboardView**: Antarmuka untuk pasien yang terdaftar.

## 2. Controller

- **AuthController**: Mengelola otentikasi pengguna, termasuk login, pendaftaran, dan pengaturan ulang password.
- **UserController**: Mengelola informasi pengguna dan interaksi dengan profil mereka.
- **DoctorController**: Mengelola data dokter dan tugas yang terkait.
- **BookingController**: Mengelola semua operasi terkait dengan janji temu, termasuk membuat dan memverifikasi booking.
- **MessageController**: Mengelola pengiriman dan penerimaan pesan antara pengguna.

## 3. Model

- **User**: Representasi pengguna dalam sistem, dengan atribut seperti **id**, **name**, **email**, dan **logs**.
- **OTP**: Menangani fitur OTP (One Time Password), menyimpan atribut untuk melakukan otentikasi melalui kode sementara.
- **Doctor**: Menyimpan informasi dokter, termasuk berbagai detail seperti **id**, **name**, dan status keberadaan mereka (**assigned**).
- **Booking**: Mengelola janji temu, termasuk atribut seperti **id**, **user\_id**, dan status booking (**confirmed**, **canceled**).
- **Message**: Menyimpan pesan yang dikirim antara dokter dan pasien, dengan atribut **id**, **message\_body**, **timestamp**, dan informasi pengirim.

## Relasi

- **User ↔ OTP**: Pengguna dapat membuat OTP untuk tujuan verifikasi.
- **User ↔ Booking**: Setiap pengguna dapat memiliki beberapa booking.
- **Doctor ↔ Booking**: Dokter dapat termasuk dalam beberapa booking.
- **User ↔ Message**: Hubungan antara pengguna dengan pesan yang mereka kirim/terima.

## BAB III

### IMPLEMENTASI SISTEM

#### 3.1. Arsitektur Sistem

**Frontend:** Dibangun menggunakan HTML, CSS, dan JavaScript untuk antarmuka pengguna (pasien dan admin).

- **Backend:** Menggunakan Node.js dengan *framework* Express.js untuk menangani logika bisnis, API, dan interaksi dengan *database*.
- **Database:** SQLite untuk menyimpan data pengguna, dokter, pemesanan, dan pesan.
- **Bot:** Integrasi dengan WhatsApp menggunakan Baileys untuk komunikasi *real-time*.

#### 3.2. Spesifikasi Perangkat Keras dan Lunak:

- **Perangkat Keras:** Server untuk *hosting* aplikasi web dan *database*  
atau komputer dengan spesifikasi berikut:
  - Linux atau Windows 10
  - RAM 1-4 GB
  - Storage 5-20 GB (semakin banyak pengguna/data maka semakin tinggi storagenya)
  - 2 Core / 2vCPU
  - Internet tergantung pemakaian, bisa ambil 5 sampai 100 GB (semakin banyak data semakin tinggi)
- **Perangkat Lunak:**
  - Sistem Operasi Server (misalnya, Linux).
  - Node.js sebagai lingkungan eksekusi *backend*.
  - Express.js sebagai *framework backend*.
  - SQLite sebagai sistem *database*.
  - Baileys sebagai *library* untuk WhatsApp Bot.

### 3.3. Front-end

#### A. UI/UX

##### 1. Halaman Login

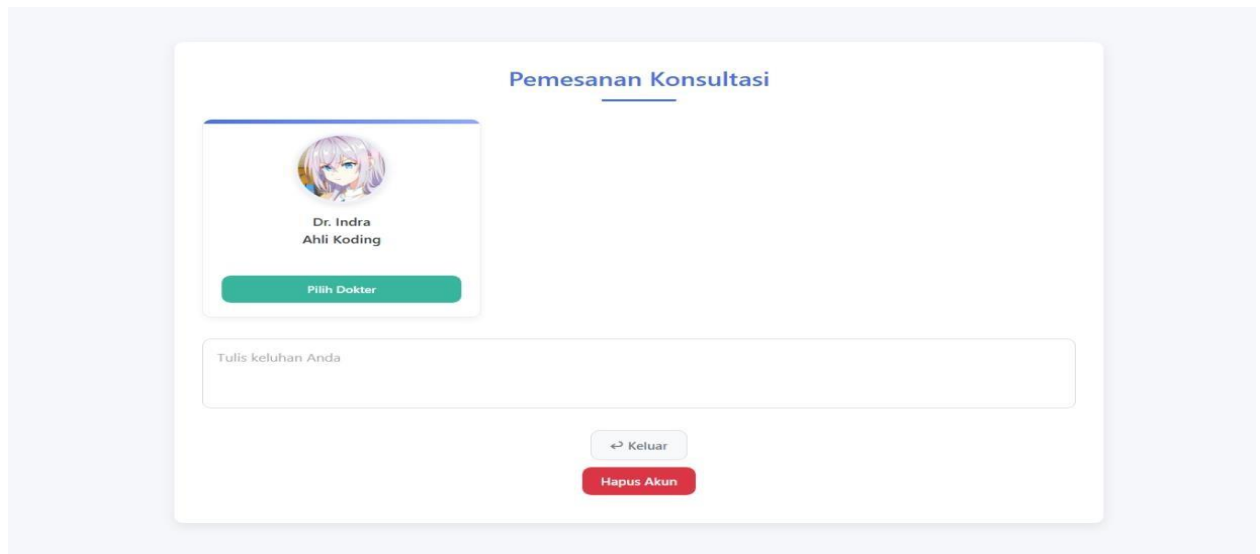
Gambar 3.1 Halaman Login

##### 2. Halaman Pendaftaran

Gambar 3.2 Halaman Pendaftaran



### 3. Dashboard User



**Pemesanan Konsultasi**

Dr. Indra  
Ahli Koding

Pilih Dokter

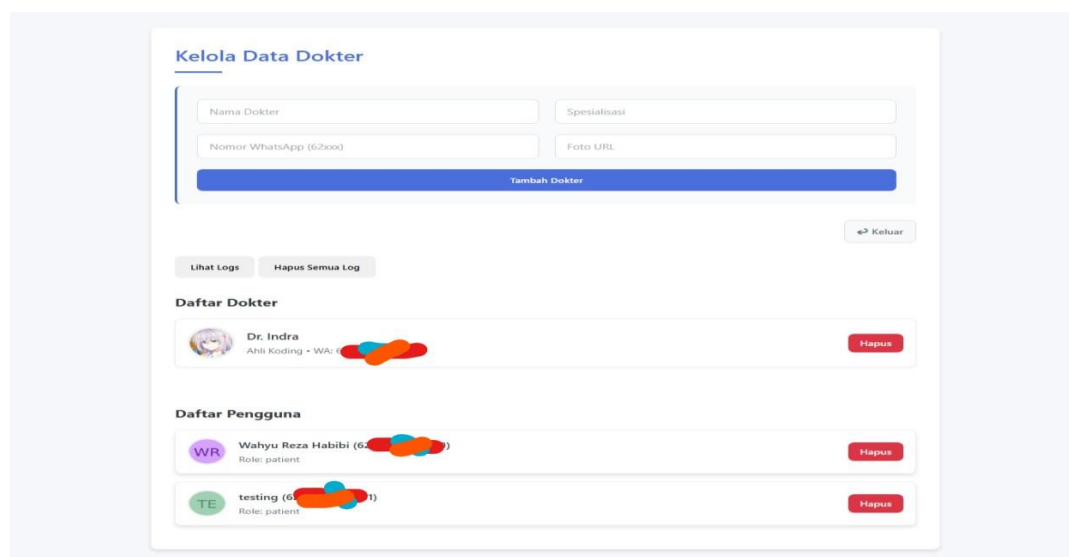
Tulis keluhan Anda

Keluar

Hapus Akun

Gambar 3.3 Dashboard User

### 4. Dashboard Admin



**Kelola Data Dokter**

Nama Dokter

Spesialisasi

Nomor WhatsApp (62xxx)

Foto URL

Tambah Dokter

Keluar

Lihat Logs

Hapus Semua Log

**Daftar Dokter**

Dr. Indra  
Ahli Koding - WA: 62xxx-xxxx-xxxx

Hapus

**Daftar Pengguna**

WR Wahyu Reza Habibi (62xxx-xxxx-xxxx)  
Role: patient

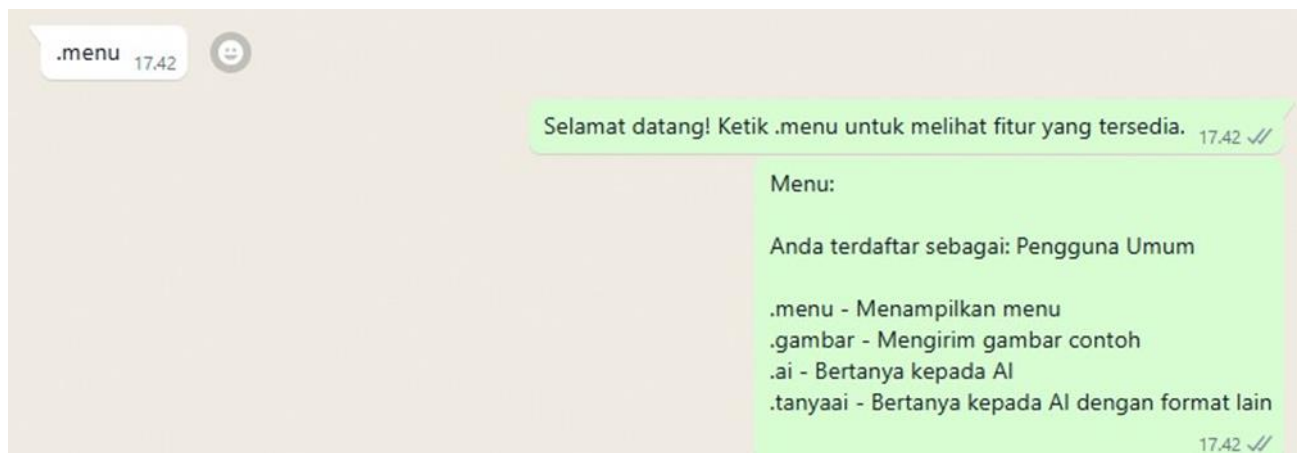
Hapus

TE testing (62xxx-xxxx-xxxx)  
Role: patient

Hapus

Gambar 3.4 Dashboard Admin

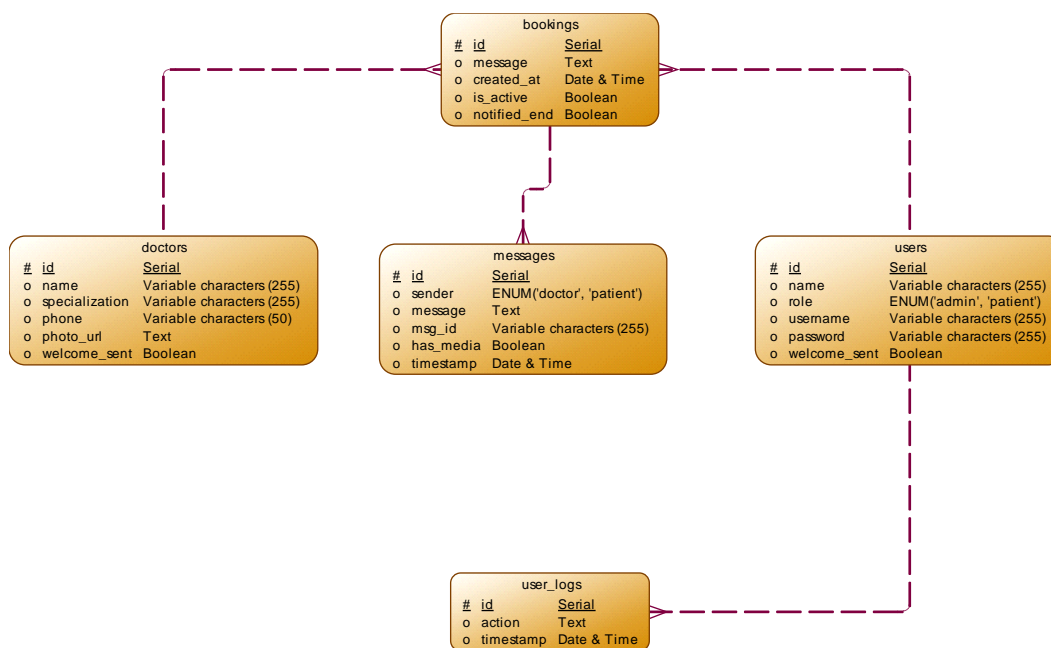
## 5. Chatbot AI



Gambar 3.5 Chatbot AI

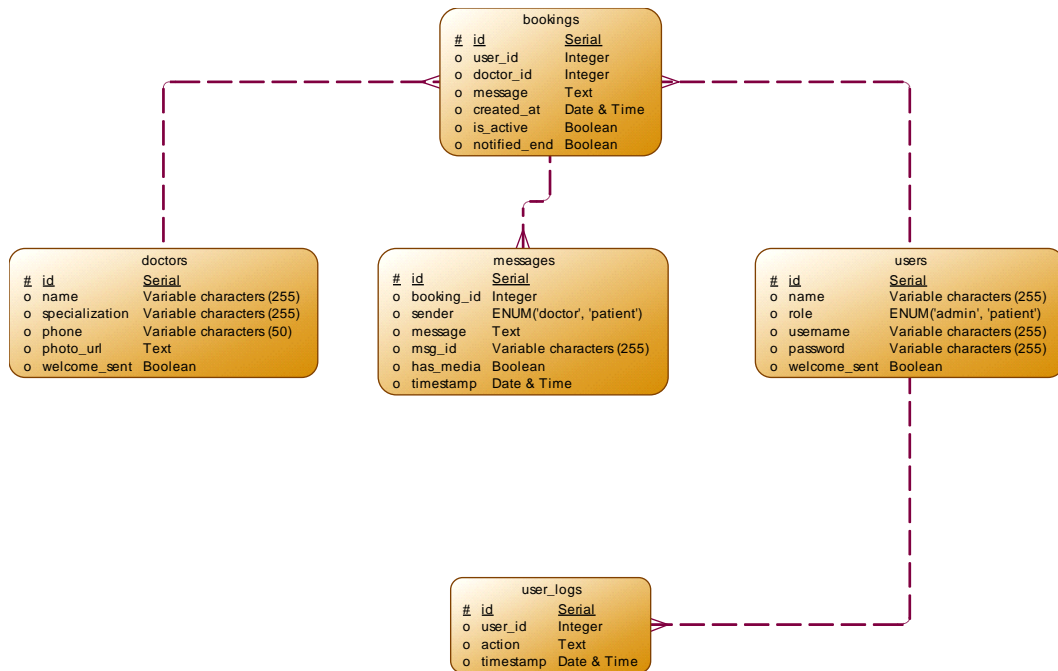
## 3.4. Back-end

### A. CDM (Conceptual Data Model)



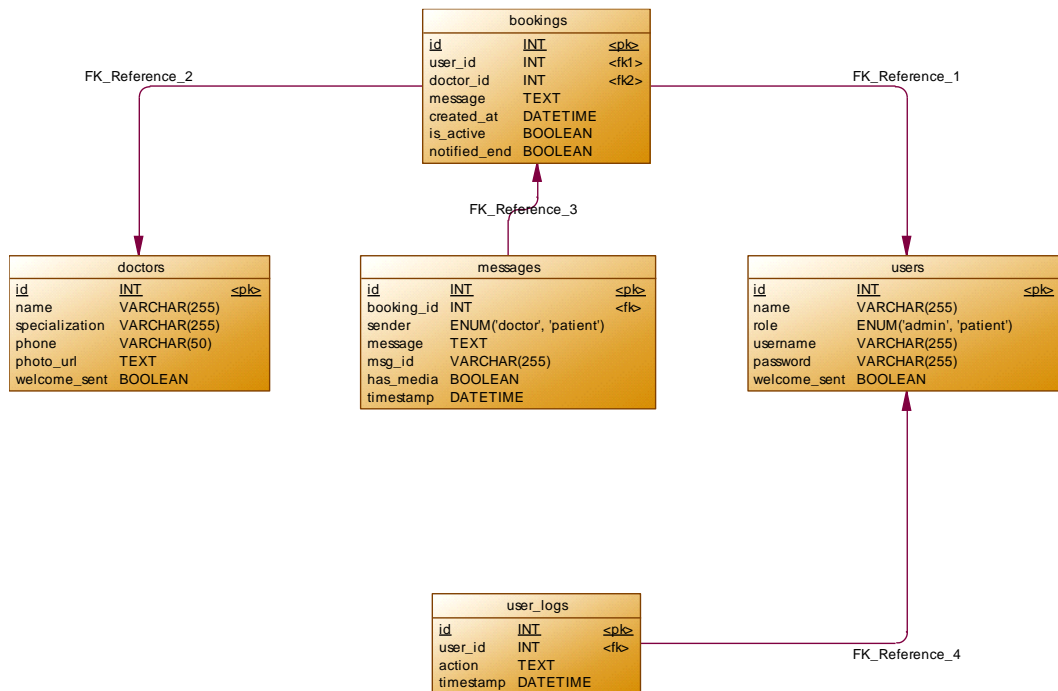
Gambar 3.6 CDM (Conceptual Data Model)

## B. LDM (Logical Data Model)



Gambar 3.7 LDM (Logical Data Model)

## C. PDM (Physical Data Model)



Gambar 3.8 PDM (Physical Data Model)

## D. SQL (Structured Query Language)

```
CREATE TABLE IF NOT EXISTS users (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  name VARCHAR(255),  
  role ENUM('admin', 'patient'),  
  username VARCHAR(255) UNIQUE,  
  password VARCHAR(255),  
  welcome_sent BOOLEAN DEFAULT FALSE  
);
```

```
CREATE TABLE IF NOT EXISTS doctors (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  name VARCHAR(255),  
  specialization VARCHAR(255),  
  phone VARCHAR(50),  
  photo_url TEXT,  
  welcome_sent BOOLEAN DEFAULT FALSE  
);
```

```
CREATE TABLE IF NOT EXISTS bookings (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  user_id INT,  
  doctor_id INT,  
  message TEXT,  
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,  
  is_active BOOLEAN DEFAULT TRUE,  
  notified_end BOOLEAN DEFAULT FALSE,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,  
  FOREIGN KEY (doctor_id) REFERENCES doctors(id) ON DELETE CASCADE  
);
```

```
CREATE TABLE IF NOT EXISTS messages (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  booking_id INT,  
  sender ENUM('doctor', 'patient'),  
  message TEXT,
```

```

msg_id VARCHAR(255),
has_media BOOLEAN,
timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (booking_id) REFERENCES bookings(id) ON DELETE CASCADE
);

CREATE TABLE IF NOT EXISTS user_logs (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT,
  action TEXT,
  timestamp DATETIME DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

```

### 3.5 Cuplikan Kode

#### 1. Kode Backend: Endpoint untuk Membuat Pemesanan (File: index.js)

Cuplikan kode ini menunjukkan bagaimana server *backend* menangani permintaan dari pasien untuk membuat sesi konsultasi baru. Kode ini menerima ID pengguna, ID dokter, dan pesan keluhan, lalu menyimpannya ke dalam *database*.

```

238 app.post('/book', isAuthenticated, (req, res) => {
239   if (!req.session.user) return res.status(403).json({ error: 'Unauthorized' });
240   const { user_id, doctor_id, message } = req.body;
241
242   // Periksa apakah pengguna sudah memiliki booking aktif
243   db.get('SELECT * FROM bookings WHERE user_id = ? AND is_active = 1', [user_id], (err, existingBooking) => {
244     if (err) return res.status(500).send('Database error');
245     if (existingBooking) {
246       return res.status(400).json({ status: 'error', message: 'Anda sudah memiliki konsultasi aktif dengan dokter lain.' });
247     }
248
249     db.run('INSERT INTO bookings (user_id, doctor_id, message) VALUES (?, ?, ?)',
250       [user_id, doctor_id, message],
251       function (err) {
252         if (err) return res.status(500).send('Database error');
253         return res.json({ status: 'success', booking_id: this.lastID });
254       });
255   });
256 });

```

## 2. Kode Bot: Penanganan Pesan Masuk (File: bot.js)

Kode ini adalah inti dari bot WhatsApp. Ia akan memeriksa setiap pesan yang masuk. Jika pesan berasal dari pasien atau dokter yang sedang dalam sesi konsultasi aktif, bot akan meneruskan pesan tersebut ke pihak yang dituju.

```
235 sock.ev.on("messages.upsert", async ({ messages }) => {
236   logger.info("Pesan baru diterima:", messages);
237   const msg = messages[0];
238   if (!msg.message || msg.key.fromMe) return;
239
240   const sender = msg.key.remoteJid;
241   const senderNumber = sender.split('@')[0];
242
243   db.get(`SELECT b.id AS booking_id, u.username, d.phone, b.is_active
244     FROM bookings b
245     JOIN users u ON b.user_id = u.id
246     JOIN doctors d ON b.doctor_id = d.id
247     WHERE u.username = ? AND b.is_active = 1
248     ORDER BY b.created_at DESC LIMIT 1`, [senderNumber], async (err, booking) => {
249     if (err) {
250       console.error("Error saat memeriksa pasien:", err);
251       return;
252     }
253
254     if (booking) {
255       // Jika pengirim adalah pasien, kirim pesan ke dokter
256       const dokterJid = `${booking.phone}@s.whatsapp.net`;
257       const replyMessage = `Ini pesan dari Nomor ${senderNumber}: ${msg.message.conversation || '[Pesan tidak tersedia]'}`;
258
259       await sock.sendMessage(dokterJid, { text: replyMessage });
260     }
261   });
262 }
```

## 3. Backend: Proses Login Pengguna dan Admin (index.js)

Cuplikan ini adalah inti dari sistem autentikasi. Kode ini menangani permintaan login, membedakan antara admin dan pengguna biasa (pasien), dan membuat sesi jika kredensial valid. Ini menunjukkan logika keamanan dasar dan interaksi dengan *database*.

```

301 // Login route
302 app.post('/login', loginLimiter, (req, res) => {
303   const { username, password } = req.body;
304
305   if (/\/s/.test(username) || /\/s/.test(password)) {
306     return res.status(400).json({ error: 'Username dan password tidak boleh mengandung spasi' });
307   }
308
309   // Hardcoded admin login
310   const adminUsername = process.env.ADMIN_USERNAME;
311   const adminPassword = process.env.ADMIN_PASSWORD;
312   if (username === adminUsername && password === adminPassword) {
313     req.session.user = { username: adminUsername, role: 'admin' };
314     return res.json({ status: 'success', role: 'admin' });
315   }
316
317   db.get('SELECT * FROM users WHERE username = ? AND password = ?', [username, password], (err, user) => {
318     if (err) {
319       console.error('Database error:', err);
320       return res.status(500).json({ error: 'Database error' });
321     }
322
323     if (user) {
324       req.session.user = { id: user.id, username: user.username, name: user.name, role: user.role };
325       logUserActivity(user.id, 'login');
326       res.json({ status: 'success', role: user.role });
327     } else {
328       res.status(401).json({ error: 'Invalid credentials' });
329     }
330   });
331 });
332

```

#### 4. Bot: Logika Penerusan Pesan antara Pasien dan Dokter (bot.js)

Ini adalah jantung dari sistem komunikasi *real-time*. Kode ini mendengarkan setiap pesan yang masuk ke nomor WhatsApp bot. Ia kemudian memeriksa apakah pengirim adalah pasien atau dokter dalam sebuah sesi aktif, lalu secara otomatis meneruskan pesan tersebut ke pihak yang dituju.

```
// Event utama yang menangani pesan masuk
sock.ev.on("messages.upsert", async ({ messages }) => {
  const msg = messages[0];
  if (!msg.message || msg.key.fromMe) return;

  const sender = msg.key.remoteJid;
  const senderNumber = sender.split('@')[0];
  const messageContent = msg.message.conversation || msg.message.extendedTextMessage?.text || "";

  // Cari sesi konsultasi yang aktif berdasarkan nomor pengirim (bisa pasien atau dokter)
  db.get(`
    SELECT b.id AS booking_id, u.username AS patient_phone, d.phone AS doctor_phone, b.is_active
    FROM bookings b
    JOIN users u ON b.user_id = u.id
    JOIN doctors d ON b.doctor_id = d.id
    WHERE (u.username = ? OR d.phone = ?) AND b.is_active = 1
    ORDER BY b.created_at DESC LIMIT 1`,
    [senderNumber, senderNumber],
    async (err, booking) => {
      if (err || !booking) return; // Abaikan jika tidak ada sesi aktif

      const pasienJid = `${booking.patient_phone}@s.whatsapp.net`;
      const dokterJid = `${booking.doctor_phone}@s.whatsapp.net`;
      let targetJid;
      let fromRole;

      // Tentukan siapa pengirim dan siapa penerima
      if (sender === dokterJid) {
        targetJid = pasienJid;
        fromRole = "doctor";
      } else if (sender === pasienJid) {
        targetJid = dokterJid;
        fromRole = "patient";
      } else {
        return;
      }

      // Teruskan pesan ke target
      await sock.sendMessage(targetJid, { text: messageContent });

      // Simpan pesan ke database
      db.run(`INSERT INTO messages (booking_id, sender, message) VALUES (?, ?, ?)`,
        [booking.booking_id, fromRole, messageContent]);
    }
  );
});
```

### 3.6 Integrasi Modul Sistem

#### 1. Pasien di Frontend:

- Pasien login dan masuk ke *dashboard* (patient.html). Halaman ini menggunakan JavaScript untuk memanggil API *backend* dan menampilkan daftar dokter yang tersedia.
- Pasien memilih dokter, menuliskan keluhan di *textarea*, lalu menekan tombol "Pilih Dokter".

#### 2. Frontend ke Backend (API Call):

- Aksi pasien memicu fungsi JavaScript yang mengirimkan permintaan **POST** ke *endpoint* /book di server *backend*. Permintaan ini membawa data *user\_id*, *doctor\_id*, dan *message* dalam format JSON.

#### 3. Backend Memproses Permintaan:



- Server **Express.js** (index.js) menerima permintaan tersebut.
- *Middleware* `isAuthenticated` akan memeriksa apakah pasien sudah login dengan valid.
- Logika di dalam *endpoint* `/book` kemudian berinteraksi dengan **Database SQLite**.

#### 4. Backend dan Database:

- *Backend* menjalankan *query* SQL untuk memeriksa apakah pasien sudah memiliki sesi aktif di tabel bookings.
- Jika tidak ada, *backend* akan menjalankan *query* INSERT untuk menyimpan data konsultasi baru ke dalam tabel bookings.

#### 5. Backend dan Bot WhatsApp:

- Setelah data berhasil disimpan, proses konsultasi resmi dimulai. **Bot WhatsApp** (bot.js), yang berjalan secara independen, secara periodik akan memeriksa *database* atau dipicu oleh *event* tertentu.
- Bot akan mendeteksi adanya sesi konsultasi baru yang belum dinotifikasi.

#### 6. Bot ke Dokter:

- Bot mengambil detail konsultasi dari *database*, termasuk nomor WhatsApp dokter dan pesan keluhan dari pasien.
- Bot kemudian mengirimkan pesan tersebut ke nomor WhatsApp dokter yang bersangkutan, menandakan bahwa ada pasien baru yang ingin berkonsultasi.

### 3.7 Panduan Instalasi dan Konfigurasi

Untuk Panduan Instalasi dan Konfigurasi ada di lampiran Silakan Ctrl + klik ini

## BAB IV

### PENGUJIAN DAN EVALUASI SISTEM

#### 4.1 Jenis Pengujian

##### 4.1.1 Whitebox Testing

##### A. Login

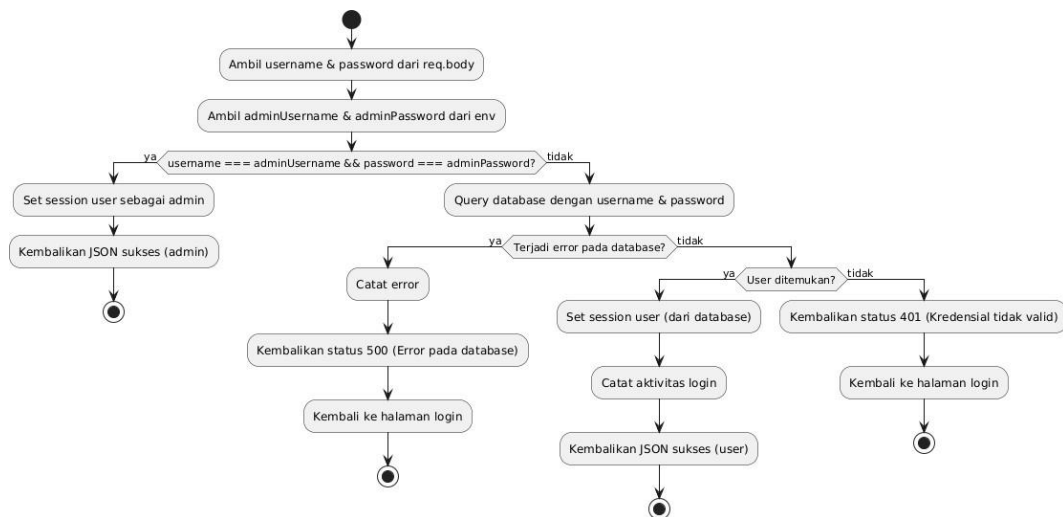
Tabel 4 1 Login

No.	Baris Kode	Deskripsi
1	<code>app.post('/login', (req, res) =&gt; {</code>	Awal route handler POST /login
2	<code>const { username, password } = req.body;</code>	Ambil username & password dari request body
3	<code>const adminUsername = process.env.ADMIN_USERNAME;</code>	Ambil admin username dari environment
4	<code>const adminPassword = process.env.ADMIN_PASSWORD;</code>	Ambil admin password dari environment
5	<code>if (username === adminUsername &amp;&amp; password === adminPassword) {</code>	Cek apakah login sebagai admin
6	<code>req.session.user = { username: adminUsername, role: 'admin' };</code>	Set sesi user sebagai admin
7	<code>return res.json({ status: 'success', role: 'admin' });</code>	Kirim respons sukses untuk admin
8	<code>}</code>	Akhir blok if admin
9	<code>db.get(SELECT * FROM users WHERE username = ? AND password = ?, [username, password], ... )</code>	Query database untuk login user biasa
10	<code>if (err) {</code>	Cek jika ada error saat akses database
11	<code>console.error('Database error:', err);</code>	Log error database
12	<code>return res.status(500).json({ error: 'Database error' });</code>	Kirim respons error 500 jika DB gagal
13	<code>}</code>	Akhir blok error
14	<code>if (user) {</code>	Cek apakah user ditemukan dari query DB
15	<code>req.session.user = { id: user.id, username: user.username, role: user.role };</code>	Set sesi user dari data user

16	logUserActivity(user.id, 'login');	Catat aktivitas login
17	res.json({ status: 'success', role: user.role });	Kirim respons sukses user biasa
18	} else {	Jika user tidak ditemukan
19	res.status(401).json({ error: 'Invalid credentials' });	Kirim respons gagal login 401
20	}	Akhir blok else user
21	});	Akhir callback DB query
22	});	Akhir route POST login


Tabel 4.2 baris yang dilewati

Path ID	Baris yang Dilewati	Keterangan
P1	1 → 2 → 3 → 4 → 5 → 6 → 7	Admin login sukses
P2	1 → 2 → 3 → 4 → 5 (false) → 9 → 10 (false) → 14 (true) → 15 → 16 → 17	User valid ditemukan di DB
P3	1 → 2 → 3 → 4 → 5 (false) → 9 → 10 (false) → 14 (false) → 18 → 19	User tidak ditemukan di DB
P4	1 → 2 → 3 → 4 → 5 (false) → 9 → 10 (true) → 11 → 12	Terjadi error saat query DB



Gambar 4.1 Login

## B. Register

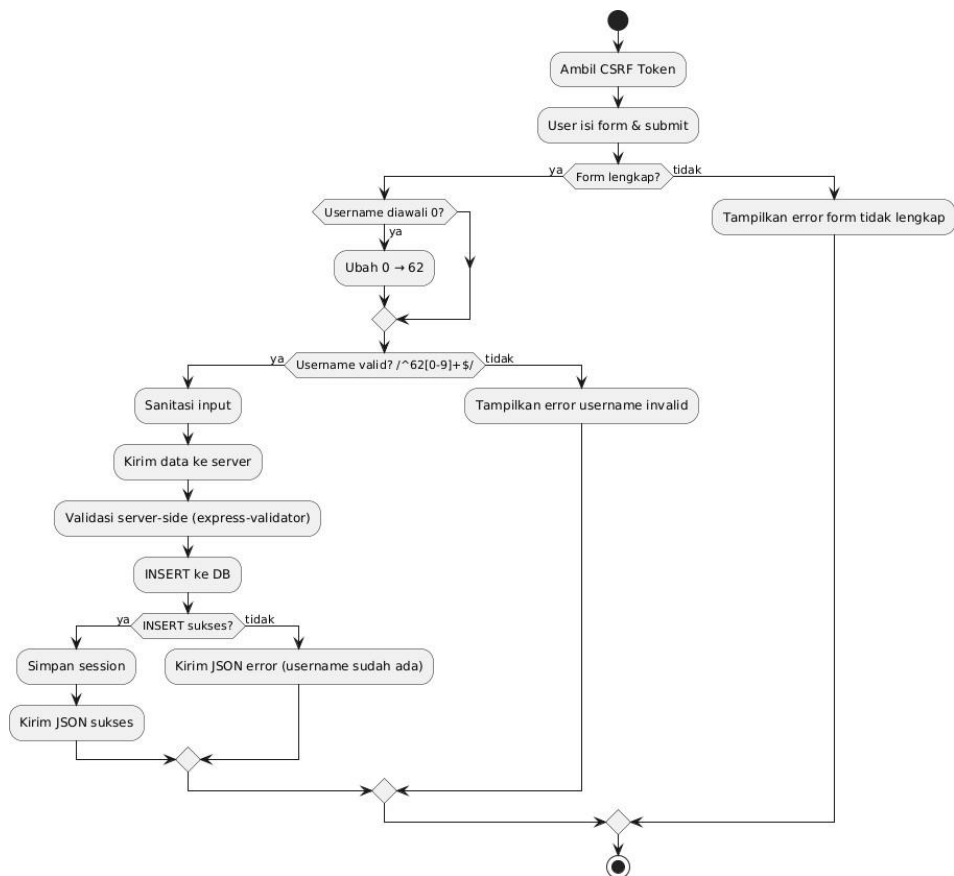
Tabel 4 3 Baris Kode

No	Baris Kode	Deskripsi
1	<code>app.post('/register', [...], (req, res))</code>	Awal route handler POST /register
2	<code>body('username').isAlphanumeric()...</code>	Validasi username: alphanumeric
3	<code>body('password').isLength({ min: 6 })...</code>	Validasi password minimal 6 karakter
4	<code>body('name').notEmpty()...</code>	Validasi name tidak boleh kosong
5	<code>(req, res) =&gt; {</code>	Callback handler
6	<code>const { username, password, name } = req.body;</code>	Ambil data pendaftaran dari request body
7	<code>console.log(Mencoba...);</code>	Log username yang akan didaftarkan
8	<code>db.run(INSERT INTO users..., [...], function (err) {</code>	Query INSERT user ke database
9	<code>if (err) {</code>	Cek apakah terjadi error di DB

10	<code>console.log('Gagal...');</code>	Log error
11	<code>return res.status(400).json({ status: 'error', message: 'Username already exists' });</code>	Kirim respons error
12	<code>console.log('Pengguna berhasil...');</code>	Log sukses pendaftaran
13	<code>req.session.user = { ... };</code>	Simpan data user di session
14	<code>res.json({ status: 'success' });</code>	Kirim respons sukses
15	<code>});</code>	Akhir callback DB
16	<code>});</code>	Akhir route handler POST /register
A1	<code>document.addEventListener('DOMContentLoaded'...</code>	Ambil CSRF token dari server
A2	<code>document.getElementById('registerForm'). addEventListener('submit'...)</code>	Tangani event submit
A3	<code>if (name === ""    username === ""    password === "") { ..... });</code>	Validasi jika ada inputan kosong
A4	<code>if (username.startsWith('0')) { ... }</code>	Validasi: jika username diawali 0 → ganti 62
A5	<code>const phoneRegex = /^62[0-9]+\$;/ if (!phoneRegex.test(username)) { ... }</code>	Validasi: username harus mulai 62 dan hanya angka
A6	<code>username, password, name = DOMPurify.sanitize(...)</code>	Sanitasi data untuk mencegah XSS / injeksi
A7	<code>fetch('/register', { method: 'POST', ... })</code>	Kirim data ke server via AJAX
A8	<code>if (data.status === 'success') { ... }</code>	Tanggapi jika server merespons sukses
A9	<code>else { ... }</code>	Tanggapi jika server merespons error

Path ID	Baris yang dilewati	Keterangan
------------	---------------------	------------

P1	A1 → A2 → A3 (false) → A4 (false) → A5 (true) → A6 → A7 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 (false) → 12 → 13 → 14	Pengguna berhasil mendaftar (username valid, data disimpan di DB)
P2	A1 → A2 → A3 (false) → A4 (false) → A5 (true) → A6 → A7 → 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 (true) → 10 → 11	Username sudah terdaftar, gagal mendaftar
P3	A1 → A2 → A3 (false) → A4 (true) → A5 (true) → A6 → A7 → 1 → 2 → 3 → 4 → ...	Nomor diawali 0, diubah jadi 62, kemudian lanjut ke proses P1/P2
P4	A1 → A2 → A3 (true)	Form tidak lengkap, ditolak di client
P5	A1 → A2 → A3 (false) → A4 (false) → A5 (false)	Username tidak valid (bukan angka 62...), ditolak di client



Gambar 4.2 Register

## A. Dashboard User atau pasien

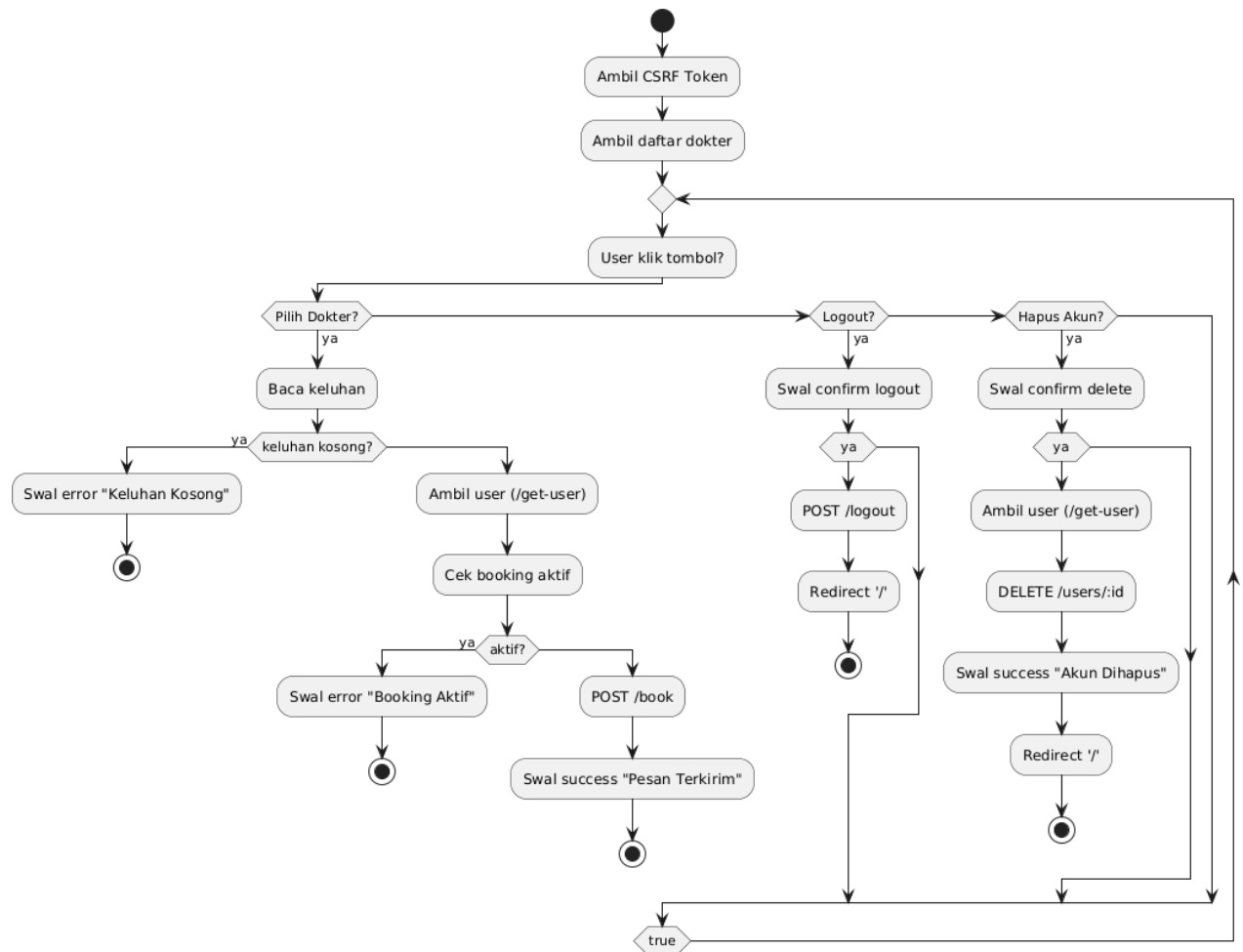
Tabel 4.4 Dashboard User atau pasien

No	Baris Kode	Deskripsi
1	<code>fetch('/csrf-token')</code>	Ambil CSRF token saat halaman dimuat
2	<code>fetch(atob(test)) // /doctors</code>	Ambil daftar dokter, sanitize dengan DOMPurify
3	<code>textarea.addEventListener('input', ...)</code>	Auto-resize textarea keluhan
4	<code>button.addEventListener('click', () =&gt; sendMessageToDoctor(doc.id))</code>	Event “Pilih Dokter” memanggil <code>sendMessageToDoctor</code>
5	<code>if (!message) { Swal.fire('Keluhan Kosong')... return; }</code>	Validasi keluhan tidak boleh kosong
6	<code>fetch('/get-user')</code>	Ambil data user login
7	<code>fetch('/bookings?user_id=\${user.id}&amp;is_active=1')</code>	Cek apakah sudah ada booking aktif
8	<code>fetch('/book', { method: 'POST', ... })</code>	Buat booking baru
9	<code>document.getElementById('deleteAccountButton').addEventListener('click', ...)</code>	Flow “Hapus Akun” → konfirmasi → ambil user → delete
10	<code>fetch('/users/\${userId}', { method: 'DELETE', ... })</code>	Kirim request hapus akun
11	<code>document.getElementById('logoutButton').addEventListener('click', ...)</code>	Flow “Logout” → konfirmasi → POST /logout
12	<code>fetch('/logout', { method: 'POST' })</code>	Kirim logout, kemudian <code>window.location.href='/'</code>

Tabel 4.5 Baris Yang Dilewati

Path ID	Baris yang Dilewati	Keterangan
P1	1 → 2 → 4 → 5 (false) → 6 → 7 (false) → 8	Booking sukses (keluhan valid, belum ada booking aktif)
P2	1 → 2 → 4 → 5 (false) → 6 → 7 (true)	Booking gagal: sudah ada booking aktif
P3	1 → 2 → 4 → 5 (true)	Booking gagal: keluhan kosong (client-side)

<b>P4</b>	1 → 9 → 6 → 10	Hapus akun sukses (konfirmasi → ambil user → delete → redirect)
<b>P5</b>	1 → 11 → 12	Logout sukses (konfirmasi → POST logout → redirect)



Gambar 4.3 Dashboard User atau pasien

### C. Dashboard Admin

Tabel 4 6 Dashboard Admin

No	Baris Kode / Fungsi	Deskripsi
1	<code>app.get('/csrf-token', (req, res) =&gt; {&lt;br&gt; // Mengembalikan token CSRF&lt;br&gt; res.json({&lt;br&gt; csrfToken: req.csrfToken() });&lt;br&gt;});</code>	Route mengembalikan CSRF token



2	<pre>app.use((err, req, res, next) =&gt; {&lt;br&gt; if (err.code === 'EBADCSRFTOKEN') {&lt;br&gt; // Token CSRF tidak valid&lt;br&gt; res.status(403).json({ error: 'Invalid CSRF token' });&lt;br&gt; } else {&lt;br&gt; next(err);&lt;br&gt; }&lt;br&gt;});</pre>	Tangani token CSRF tidak valid (403)
3	<pre>app.get('/', (req, res) =&gt; {&lt;br&gt; res.sendFile(path.join(__dirname, 'public/index.html'));&lt;br&gt;});</pre>	Kirim index.html
4	<pre>app.use((err, req, res, next) =&gt; {&lt;br&gt; console.error(err);&lt;br&gt; res.status(500).json({ message: 'Internal Server Error' });&lt;br&gt;});</pre>	Tangani semua error tak terduga (500)
5	<pre>app.get('/admin/logs', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; db.all('SELECT l.id, u.username, l.action, l.timestamp&lt;br&gt; FROM user_logs l&lt;br&gt; JOIN users u ON l.user_id = u.id&lt;br&gt; ORDER BY l.timestamp DESC', [], (err, rows) =&gt; {&lt;br&gt; if (err) {&lt;br&gt; return res.status(500).json({ error: 'Gagal mengambil log aktivitas' });&lt;br&gt; }&lt;br&gt; res.json(rows);&lt;br&gt; });&lt;br&gt;});</pre>	Ambil semua log aktivitas (JOIN users + user_logs)
6	<pre>app.delete('/admin/logs', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; db.run('DELETE FROM user_logs', [], function (err) {&lt;br&gt; if (err) {&lt;br&gt; console.error('Error deleting logs:', err);&lt;br&gt; return res.status(500).json({ error: 'Gagal menghapus log aktivitas' });&lt;br&gt; }&lt;br&gt; res.json({ status: 'success', message: 'Semua log aktivitas berhasil dihapus' });&lt;br&gt; });&lt;br&gt;});</pre>	Hapus seluruh log aktivitas
7	<pre>app.get('/doctors', isAuthenticated, (req, res) =&gt; {&lt;br&gt; if (!req.session.user) return res.status(403).json({ error: 'Unauthorized' });&lt;br&gt; db.all('SELECT * FROM doctors', (err, rows) =&gt; {&lt;br&gt; if (err) return res.status(500).json({ error: 'Database error' });&lt;br&gt; res.json(rows);&lt;br&gt; });&lt;br&gt;});</pre>	Ambil daftar dokter (cek session, DB query)
8	<pre>app.post('/doctors', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; const { name, specialization, phone, photo_url } = req.body;&lt;br&gt; if (!photo_url.startsWith('http://') &amp;&amp; !photo_url.startsWith('https://')) {&lt;br&gt; return res.status(400).json({ error: 'Invalid photo URL' });&lt;br&gt; }&lt;br&gt; db.run('INSERT INTO doctors (name, specialization, phone, photo_url) VALUES (?, ?, ?, ?)',&lt;br&gt; [name, specialization, phone, photo_url], err =&gt; {&lt;br&gt; if (err) return res.status(500).json({ error: 'Gagal tambah dokter' });&lt;br&gt; res.json({ status: 'success' });&lt;br&gt; });&lt;br&gt;});</pre>	Tambah dokter baru (validasi URL foto, DB INSERT)
9	<pre>app.delete('/doctors/:id', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; const id = req.params.id;&lt;br&gt; db.run('DELETE FROM doctors WHERE id = ?', [id], function(err) {&lt;br&gt; if (err) return res.status(500).json({ error: 'Gagal hapus dokter' });&lt;br&gt; res.json({ status: 'deleted' });&lt;br&gt; });&lt;br&gt;});</pre>	Hapus dokter berdasarkan ID

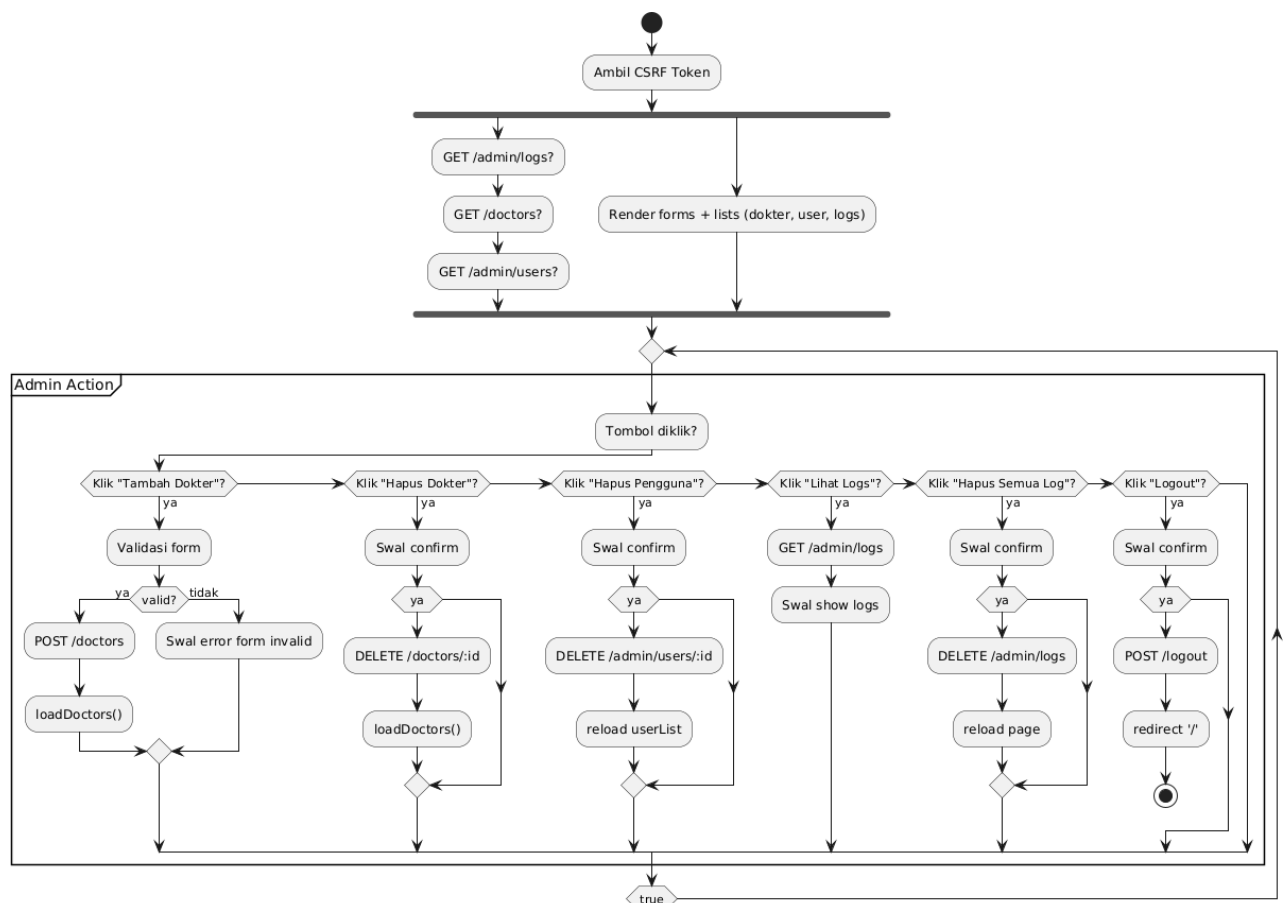
10	<pre>app.delete('/admin/users/:id', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; const userId = req.params.id;&lt;br&gt; db.run('DELETE FROM users WHERE id = ?', [userId], function (err) {&lt;br&gt; if (err) return res.status(500).json({ error: 'Gagal menghapus akun pengguna' });&lt;br&gt; logUserActivity(`\${userId}`, 'delete_account_by_admin');&lt;br&gt; res.json({ status: 'success', message: `Akun dengan ID \${userId} berhasil dihapus` });&lt;br&gt; });&lt;br&gt; });</pre>	Hapus user oleh admin (DB DELETE + logUserActivity)
11	<pre>app.get('/admin/users', isAuthenticated &amp;&amp; isAdmin, (req, res) =&gt; {&lt;br&gt; db.all('SELECT id, username, name, role FROM users', (err, rows) =&gt; {&lt;br&gt; if (err) return res.status(500).json({ error: 'Gagal mendapatkan daftar pengguna' });&lt;br&gt; res.json(rows);&lt;br&gt; });&lt;br&gt; });</pre>	Ambil daftar semua user
12	<pre>app.post('/logout', (req, res) =&gt; {&lt;br&gt; logUserActivity(req.session.user.id, 'logout');&lt;br&gt; req.session.destroy(err =&gt; {&lt;br&gt; if (err) return res.status(500).json({ error: 'Gagal logout' });&lt;br&gt; res.json({ status: 'success', message: 'Logged out successfully' });&lt;br&gt; });&lt;br&gt; });</pre>	Logout user (destroy session + logUserActivity)
13	<pre>fetch('/csrf-token')&lt;br&gt; .then(res =&gt; res.json())&lt;br&gt; .then(data =&gt; { document.getElementById('csrfToken').value = data.csrfToken; });</pre>	Ambil dan simpan CSRF token di #csrfToken
14	<pre>fetch('/admin/logs')&lt;br&gt; .then(res =&gt; res.json())&lt;br&gt; .then(data =&gt; Swal.fire({ html: logsHtml }));&lt;br&gt;&lt;br&gt;fetch('/admin/logs', { method: 'DELETE', headers: { 'X-CSRF-Token': csrfToken } })</pre>	Flow “Lihat Logs” & “Hapus Semua Log” (Swal confirm + fetch)

15	<pre>function loadDoctors() {&lt;br&gt; fetch('/doctors')&lt;br&gt; .then(res =&gt; res.json())&lt;br&gt; .then(data =&gt; { /* render list + tombol Hapus */ });&lt;br&gt; }</pre>	Render daftar dokter, tombol hapus dokter
16	<pre>document.getElementById('addDoctorForm').addEventListener('submit', e =&gt; {&lt;br&gt; e.preventDefault();&lt;br&gt; fetch('/doctors', { method: 'POST', headers: { 'X-CSRF-Token': csrfToken }, body: JSON.stringify({ name, specialization, phone, photo_url }) })&lt;br&gt; });</pre>	Tambah dokter via form (validasi HTML5 + Swal)

17	function deleteDoctor(id) {  fetch(`/doctors/\${id}`, { method: 'DELETE', headers: { 'X-CSRF-Token': csrfToken } })  .then(() => loadDoctors()); }	Hapus dokter via button “Hapus” + CSRF header
18	fetch('/admin/users')  .then(res => res.json())  .then(users => { /* render userList + tombol Hapus */ });	Tampilkan daftar user + tombol hapus pengguna
19	function deleteUser(userId) {  fetch(`/admin/users/\${userId}`, { method: 'DELETE', headers: { 'X-CSRF-Token': csrfToken } })  .then(() => location.reload()); }	Hapus user via Swal confirm + CSRF header

Tabel 4 7Baris Yang Dilewai

Path ID	Baris yang Dilewati	Keterangan
P1	1 → 3 → 7 → 15 → 16	Tambah dokter sukses (GET csrf, loadDoctors, submit form valid)
P2	1 → 3 → 7 → 15 → 17	Hapus dokter sukses (GET csrf, loadDoctors, Swal confirm, DELETE)
P3	1 → 2	Error CSRF invalid (middleware)
P4	1 → 3 → 5 → 14	Lihat logs sukses (GET logs, render in Swal)
P5	1 → 3 → 6 → 14	Hapus semua log sukses (GET csrf, Swal confirm, DELETE logs)
P6	1 → 3 → 11 → 18	Tampil userList sukses (GET csrf, GET admin/users, render)
P7	1 → 3 → 10 → 18	Hapus user oleh admin (GET csrf, Swal confirm, DELETE admin/users)
P8	1 → 3 → 12	Logout sukses (GET csrf, Swal confirm, POST logout, redirect)
P9	1 → 4	Error global (500) downstream route



Gambar 4.4 Dashboard Admin

## D. OTP (One Time Password)

Tabel 4 8 OTP (One Time Password)

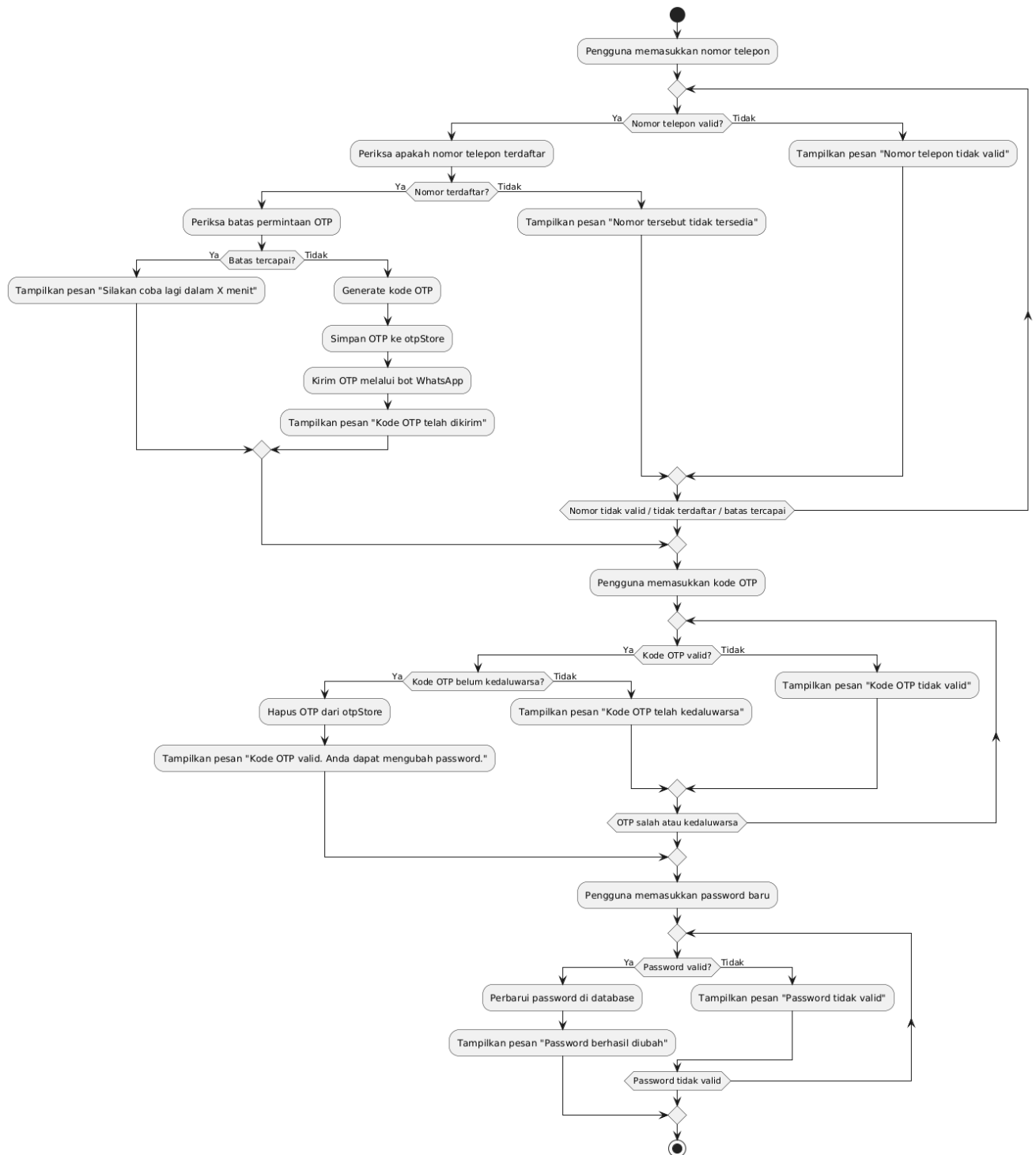
No.	Baris Kode / Fungsi	Deskripsi
1	<code>app.post('/forgot-password/send-otp', [...], async (req, res) =&gt; {...});</code>	Route untuk mengirim OTP ke nomor telepon
2	<code>body('phone')...</code> dan validasi manual phone	Validasi nomor telepon (format, panjang, kosong)
3	<code>db.get('SELECT * FROM users WHERE username = ?', [phone], ...)</code>	Cek apakah nomor telepon terdaftar
4	<code>if (count &gt;= 3) {...}</code>	Cek apakah sudah melewati batas request OTP
5	<code>const otp = crypto.randomInt(...)</code>	Generate OTP acak 6 digit

6	otpStore[phone] = { otp, expiresAt: ... }	Simpan OTP dan waktu kedaluwarsa
7	await sendOtpViaBot(phone, otp);	Kirim OTP ke WhatsApp melalui bot
8	app.post('/forgot-password/verify-otp', [...], (req, res) => {...});	Route untuk verifikasi OTP
9	if (!storedOtp)	Cek apakah OTP ada di memory
10	if (storedOtp.otp !== parseInt(otp))	Cek apakah OTP cocok
11	if (Date.now() > storedOtp.expiresAt)	Cek apakah OTP sudah kedaluwarsa
12	delete otpStore[phone];	Hapus OTP dari memory setelah verifikasi sukses
13	app.post('/forgot-password/change-password', [...], (req, res) => {...});	Route untuk ubah password
14	body('newPassword')... dan validasi manual	Validasi password baru (kosong, panjang, spasi)
15	db.run('UPDATE users SET password = ? WHERE username = ?', [...])	Update password pengguna di database
16	Swal.fire('...', '...', '...') di JS frontend	Menampilkan feedback ke pengguna
17	fetch('/forgot-password/send-otp', ...)	Mengirim permintaan OTP dari frontend
18	fetch('/forgot-password/verify-otp', ...)	Mengirim OTP yang dimasukkan user ke backend
19	fetch('/forgot-password/change-password', ...)	Mengirim password baru ke backend

Tabel 4.9 Baris yang dilewati

Path ID	Baris yang Dilewati	Keterangan
P1	1 → 2 → 3 (terdaftar) → 4 (limit belum tercapai) → 5 → 6 → 7 → 8 → 9 → 10 (OTP cocok) → 11 (tidak expired) → 12 → 13 → 14 (password valid) → 15	Jalur sukses penuh: kirim OTP, verifikasi berhasil, ubah password berhasil

<b>P2</b>	1 → 2 (validasi gagal)	Nomor telepon tidak valid
<b>P3</b>	1 → 2 → 3 (TIDAK terdaftar)	Nomor tidak ditemukan di database
<b>P4</b>	1 → 2 → 3 → 4 (limit tercapai)	Gagal karena melebihi batas permintaan OTP
<b>P5</b>	8 → 9 (OTP tidak ada)	OTP belum diminta atau sudah expired
<b>P6</b>	8 → 9 → 10 (OTP salah)	OTP tidak cocok
<b>P7</b>	8 → 9 → 10 → 11 (OTP expired)	OTP sudah kedaluwarsa
<b>P8</b>	13 → 14 (validasi gagal - password terlalu pendek / spasi / panjang / kosong)	Gagal ubah password karena input invalid
<b>P9</b>	13 → 14 → 15	Berhasil ubah password setelah validasi



Gambar 4.5 OTP (One Time Password)

#### 4.1.2 Blackbox Testing

##### A. Login

*Tabel 4 10 Login*

Identitas Pengujian	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Login (Valid)	Pengguna login dengan username dan password yang benar	Tidak dapat login	Masuk	Berhasil
Login (Invalid)	Pengguna login dengan password yang salah	Pengguna menekan tombol daftar tanpa mengisi form	Muncul Pesan	Berhasil

##### B. Register

*Tabel 4 11 Register*

Identitas Pengujian	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Register	Pengguna membuat akun dengan memasukan username, Nomer HP, dan Password	Akun berhasil dibuat dan masuk kedalam website	Masuk	Berhasil
Register (Kosong)	Pengguna menekan tombol daftar tanpa mengisi form	Pengguna menekan tombol daftar tanpa mengisi form	Muncul Pesan	Berhasil
Register (Duplikat)	Pengguna daftar dengan username/Nomor HP yang sudah terdaftar	Muncul pesan "sudah terdaftar"	Muncul Pesan	Berhasil



### C. Dashboard User atau pasien

Tabel 4 12 Dashboard User atau pasien

Identitas Pengujian	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Pilih Dokter	Pengguna menekan tombol "Pilih Dokter"	Dokter terpilih dan data muncul/tersimpan	Dokter Terpilih	Berhasil
Tulis Keluhan	Pengguna mengisi kolom teks keluhan	Keluhan berhasil ditulis dan disimpan/kirim	Keluhan Tersimpan	Berhasil
Tulis Keluhan (Kosong)	Pengguna klik kirim tanpa mengisi kolom keluhan	Muncul pesan peringatan "kolom wajib diisi"	Muncul Pesan	Berhasil
Logout	Pengguna menekan tombol "Keluar"	Keluar dan kembali ke halaman login/home	Kembali Ke Login	Berhasil
Hapus Akun (konfirmasi)	Pengguna klik "Hapus Akun" dan menekan "Ya" saat konfirmasi	Akun terhapus dan diarahkan ke halaman awal	Akun Terhapus	Berhasil
Hapus Akun (Batal)	Pengguna klik "Hapus Akun" lalu memilih "Batal" di dialog konfirmasi	Proses batal, tetap di dashboard	Tidak Terhapus	Berhasil

#### D. Dashboard Admin

Tabel 4.13 Dashboard Admin

Identitas Pengujian	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Tambah Dokter (Valid Input)	Admin mengisi seluruh field dengan data yang benar lalu klik "Tambah Dokter"	Dokter ditambahkan ke daftar	Dokter Muncul	Berhasil
Tambah Dokter (Kosong)	Admin tidak mengisi satu atau beberapa field lalu klik "Tambah Dokter"	Muncul pesan error bahwa semua field wajib diisi	Muncul Pesan	Berhasil
Tambah Dokter (Nomor Tidak Valid)	Admin mengisi nomor WhatsApp tanpa format 62 atau dengan huruf	Muncul peringatan format salah	Muncul Pesan	Berhasil

Hapus Dokter	Admin klik tombol "Hapus" di salah satu data dokter	Data dokter tersebut dihapus dari daftar	Dokter Terhapus	Berhasil
Hapus Pengguna	Admin klik tombol "Hapus" di salah satu data pengguna	Data pengguna terhapus dari daftar	Pengguna Terhapus	Berhasil
Lihat Log	Admin klik tombol "Lihat Log"	Log aktivitas ditampilkan	Log Tampil	Berhasil
Hapus Semua Log	Admin klik tombol "Hapus Semua Log"	Semua log terhapus dan list kosong	Log Terhapus	Berhasil
Logout (Keluar)	Admin klik tombol "Keluar"	Sistem keluar ke halaman login atau utama	Kembali ke login	Berhasil

## E. OTP (One Time Password)

Tabel 4 14 OTP (One Time Password)

Identitas Pengujian	Deskripsi	Hasil Yang Diharapkan	Hasil Pengujian	Kesimpulan
Kirim OTP (Valid)	Pengguna memasukkan nomor telepon yang valid dan terdaftar	Kode OTP terkirim ke nomor telepon	Kode OTP terkirim ke nomor telepon	Berhasil
Kirim OTP (Invalid)	Pengguna memasukkan nomor telepon yang tidak valid atau tidak terdaftar	Muncul pesan "Nomor tersebut tidak tersedia"	Muncul pesan "Nomor tersebut tidak tersedia"	Berhasil
Kirim OTP (Limit)	Pengguna telah mencapai batas permintaan OTP (3 kali)	Muncul pesan "Silakan coba lagi dalam X menit"	Muncul pesan "Silakan coba lagi dalam X menit"	Berhasil
Verifikasi OTP (Valid)	Pengguna memasukkan kode OTP yang benar	Muncul pesan "Kode OTP valid. Anda dapat mengubah password."	Muncul pesan "Kode OTP valid. Anda dapat mengubah password."	Berhasil
Verifikasi OTP (Invalid)	Pengguna memasukkan kode OTP yang salah	Muncul pesan "Kode OTP tidak valid."	Muncul pesan "Kode OTP tidak valid."	Berhasil
Verifikasi OTP (Expired)	Pengguna memasukkan kode OTP yang telah kedaluwarsa	Muncul pesan "Kode OTP telah kedaluwarsa."	Muncul pesan "Kode OTP telah kedaluwarsa."	Berhasil
Ubah Password (Valid)	Pengguna memasukkan password baru yang valid	Password berhasil diubah	Password berhasil diubah	Berhasil
Ubah Password (Invalid)	Pengguna memasukkan password baru yang tidak	Muncul pesan "Password tidak valid."	Muncul pesan "Password tidak valid."	Berhasil

	valid (misalnya terlalu pendek)			
--	---------------------------------	--	--	--

## 4.2 Hasil Pengujian dan Analisis

Pada tahap ini, dilakukan analisis mendalam terhadap hasil pengujian *White-box* dan *Black-box* untuk mengevaluasi fungsionalitas, alur logika, dan kesesuaian sistem dengan kebutuhan pengguna. Analisis ini bertujuan untuk memastikan bahwa setiap komponen aplikasi "Medical Conversation" berjalan sesuai dengan yang diharapkan.

### 4.2.1 Pengujian White-box dan Analisis

Pengujian *White-box* difokuskan pada pengujian struktur internal dan alur logika dari kode sumber. Analisis dilakukan dengan menelusuri berbagai jalur eksekusi kode untuk setiap fungsionalitas utama.

## 1. Fungsionalitas Login

- **Analisis:**
  - **Jalur Sukses (P1 & P2):** Pengujian menunjukkan bahwa logika untuk membedakan login admin dan login pengguna biasa berjalan dengan benar. Untuk admin, sistem melakukan perbandingan dengan variabel lingkungan, sedangkan untuk pengguna, sistem melakukan *query* ke tabel *users* di *database*. Jika berhasil, sesi pengguna akan dibuat (*req.session.user*) yang berisi peran dan ID pengguna, yang menjadi dasar otorisasi di seluruh aplikasi.
  - **Jalur Gagal (P3):** Skenario ini berhasil membuktikan bahwa jika *query* ke *database* tidak menemukan kombinasi *username* dan *password* yang cocok, sistem akan masuk ke blok *else* dan

mengembalikan respons 401 Invalid credentials. Ini penting untuk mencegah akses yang tidak sah.

- **Jalur Error (P4):** Jalur ini menguji penanganan kesalahan saat terjadi masalah pada koneksi atau *query* ke *database*. Sistem terbukti mampu menangkap *error* (if (err)) dan mengembalikan status 500 Database error, yang mencegah aplikasi dari *crash* dan memberikan umpan balik yang jelas kepada *developer*.

## 2. Fungsionalitas Pendaftaran (Register)

- **Analisis:**

- **Jalur Sukses (P1):** Jalur ini memvalidasi alur pendaftaran yang ideal. Di sisi *frontend* (register.html), JavaScript melakukan validasi awal seperti format nomor telepon (^62[0-9]+\$) dan sanitasi input menggunakan DOMPurify untuk mencegah serangan XSS. Di sisi *backend* (index.js), *middleware* express-validator melakukan validasi ulang sebelum data dimasukkan ke *database* melalui perintah INSERT INTO users. Jika berhasil, sesi pengguna langsung dibuat.
- **Jalur Gagal karena Duplikat (P2):** Pengujian membuktikan bahwa *database* memiliki batasan UNIQUE pada kolom username. Ketika terjadi duplikasi, perintah db.run akan menghasilkan *error* (if (err)), yang kemudian ditangkap oleh logika *backend* untuk mengembalikan status 400 dengan pesan "Username already exists".
- **Jalur Gagal di Frontend (P4 & P5):** Skenario ini menunjukkan bahwa validasi di sisi klien berfungsi efektif. Jika pengguna tidak mengisi semua kolom atau memasukkan format nomor telepon yang salah, permintaan tidak akan pernah dikirim ke server, dan pengguna akan menerima umpan balik langsung melalui Swal.fire.

## 3. Fungsionalitas Dashboard Pasien

- **Analisis:**

- **Jalur Sukses Booking (P1):** Alur ini berhasil memvalidasi proses pemesanan konsultasi. Logika *frontend* (patient.html) pertama-tama akan memeriksa apakah ada sesi aktif dengan melakukan fetch ke *endpoint* /bookings. Jika tidak ada, permintaan POST akan dikirim ke /book. *Backend* kemudian akan menyimpan data ini ke *database*, yang menandakan dimulainya sesi konsultasi.
- **Jalur Gagal Booking (P2):** Skenario ini membuktikan bahwa sistem dapat mencegah pemesanan ganda. Jika fetch awal menemukan adanya sesi aktif, *frontend* akan menampilkan pesan error "Booking Aktif" dan tidak akan melanjutkan proses pengiriman data.
- **Jalur Logout dan Hapus Akun (P4 & P5):** Kedua alur ini berhasil divalidasi. Prosesnya melibatkan konfirmasi dari pengguna melalui Swal.fire, diikuti dengan pengiriman permintaan POST /logout atau DELETE /users/:id ke *backend*. *Backend* kemudian akan menghancurkan sesi (req.session.destroy()) atau menghapus data dari *database*.

#### 4. Fungsionalitas Lupa Password (OTP)

- **Analisis:**
  - **Jalur Sukses Penuh (P1):** Alur ini memvalidasi keseluruhan proses dari pengiriman OTP hingga perubahan *password*. Proses dimulai saat *backend* menerima permintaan kirim OTP, memvalidasi nomor telepon di *database*, menghasilkan OTP acak, menyimpannya sementara di otpStore, dan mengirimkannya melalui sendOtpViaBot. Setelah verifikasi berhasil, OTP akan dihapus dari otpStore dan *password* diubah di *database* menggunakan perintah UPDATE.
  - **Jalur Gagal (P2, P3, P4):** Pengujian membuktikan bahwa sistem memiliki beberapa lapisan validasi. Jika nomor tidak valid atau tidak terdaftar, sistem akan mengembalikan pesan error. Sistem juga berhasil menerapkan *rate limiting* untuk mencegah *spam* OTP, di

mana jika batas permintaan terlampaui, pengguna harus menunggu sebelum bisa meminta lagi.

- **Jalur Verifikasi Gagal (P5, P6, P7):** Skenario ini menunjukkan bahwa logika verifikasi di *backend* berjalan dengan baik. Sistem akan memeriksa apakah OTP yang dimasukkan cocok dengan yang ada di *otpStore* dan apakah OTP tersebut belum kedaluwarsa. Jika salah satu kondisi tidak terpenuhi, proses akan gagal.

#### 4.2.2 Pengujian Black-box dan Analisis

Pengujian *Black-box* dilakukan dari perspektif pengguna akhir tanpa melihat struktur kode internal. Tujuannya adalah untuk memverifikasi bahwa setiap fitur berfungsi sesuai dengan kebutuhan fungsional yang telah ditentukan.

##### 1. Fungsionalitas Login

- **Hasil Pengujian:**
  - **Login Valid:** Pengguna berhasil masuk ke sistem saat menggunakan kredensial yang benar.
  - **Login Invalid:** Sistem menampilkan pesan kesalahan dan menolak akses saat pengguna memasukkan kredensial yang salah.
- **Analisis Hasil:** Hasil pengujian *Black-box* untuk fungsionalitas login dinyatakan **Berhasil**. Sistem mampu membedakan input yang valid dan invalid, serta memberikan respons yang sesuai, sejalan dengan spesifikasi kebutuhan fungsional.

##### 2. Fungsionalitas Pendaftaran (Register)

- **Hasil Pengujian:**

- **Register Valid:** Pengguna berhasil membuat akun baru dan masuk ke *dashboard*.
- **Register Kosong:** Sistem menampilkan pesan peringatan saat pengguna mencoba mendaftar dengan form kosong.
- **Register Duplikat:** Sistem menampilkan pesan bahwa *username* sudah terdaftar saat pengguna mencoba mendaftar dengan nomor HP yang sama.
- **Analisis Hasil:** Fungsionalitas pendaftaran dinyatakan **Berhasil**. Sistem telah terbukti mampu menangani skenario pendaftaran normal, validasi input kosong, dan pencegahan duplikasi akun, yang semuanya merupakan kebutuhan kritis.

### 3. Fungsionalitas Dashboard Pasien

- **Hasil Pengujian:**
  - **Pilih Dokter dan Tulis Keluhan:** Keluhan pengguna berhasil terkirim dan tersimpan saat dokter dipilih.
  - **Tulis Keluhan Kosong:** Sistem menampilkan pesan peringatan saat pengguna mencoba mengirim keluhan tanpa mengisinya.
  - **Logout dan Hapus Akun:** Fungsi keluar dan hapus akun berjalan sesuai harapan, dengan konfirmasi yang memastikan tindakan tidak dilakukan secara tidak sengaja.
- **Analisis Hasil:** Fungsionalitas *dashboard* pasien dinyatakan **Berhasil**. Semua interaksi pengguna, mulai dari pengiriman keluhan hingga manajemen akun, memberikan hasil yang diharapkan dan menunjukkan bahwa alur kerja pengguna telah terimplementasi dengan baik.

### 4. Fungsionalitas Dashboard Admin

- **Hasil Pengujian:**



- **Manajemen Dokter:** Admin berhasil menambah dan menghapus data dokter. Validasi input untuk form tambah dokter juga berfungsi dengan baik.
- **Manajemen Pengguna:** Admin berhasil menghapus data pengguna dari sistem.
- **Manajemen Log:** Admin dapat melihat dan menghapus log aktivitas sistem.
- **Analisis Hasil:** Fungsionalitas *dashboard* admin dinyatakan **Berhasil**. Sistem menyediakan kontrol administratif yang diperlukan dan semua fungsi berjalan sesuai spesifikasi, membuktikan bahwa hak akses dan fitur-fitur khusus admin telah diimplementasikan dengan benar.

## 5. Fungsionalitas Lupa Password (OTP)

- **Hasil Pengujian:**
  - **Alur OTP Valid:** Pengguna berhasil menerima OTP, melakukan verifikasi, dan mengubah *password* mereka.
  - **Alur OTP Invalid:** Sistem dengan benar menangani semua skenario negatif, seperti nomor tidak terdaftar, OTP salah, OTP kedaluwarsa, dan permintaan OTP yang berlebihan.
- **Analisis Hasil:** Fungsionalitas OTP dinyatakan **Berhasil**. Sistem terbukti andal dan aman dalam menangani proses pemulihan akun, yang merupakan salah satu fitur keamanan paling vital dalam aplikasi.

## 4.3 Evaluasi Sistem oleh Pengguna

Evaluasi sistem oleh pengguna dilakukan melalui sesi tanya jawab dan demonstrasi aplikasi. Sesi ini bertujuan untuk mengumpulkan umpan balik, mengidentifikasi potensi kebingungan, dan mengukur pemahaman pengguna terhadap alur kerja aplikasi "Medical Conversation". Berikut adalah rekapitulasi dari evaluasi yang dilakukan berdasarkan pertanyaan-pertanyaan yang diajukan:

Tabel 4 15 Pertanyaan

No	Kelompok	Pertanyaan	Jawaban
1	5	Jika user atau pengguna lupa kata sandi itu bagaimana ?	Jika tidak ada kata sandi, bisa menggunakan kontak Instagram atau sosial media lainnya ke admin atau bisa menggunakan kode otp.
2	4	Feedback atau umpan balik dari dokter tidak terlihat di user ?	Karena feedback dari dokter tersebut itu dari WhatsApp, jadi setelah dari website yang mengirim keluhan lalu diteruskan ke dokter, dan dokter menerima pesan, dan dokter membalas, pada sisi bot, bot mengirimkan pesan yang dikirim dari dokter untuk diteruskan ke pasien yang sudah daftar di website menggunakan nomor WhatsApp yang valid atau aktif atau benar
3	6	kenapa menggunakan admin.html ?	memang benar menggunakan admin.html akan tetapi pada bagian backend atau ExpressJS ini, kita me-routing admin.html dan pasien.html ini ke /dashboard yang dimana admin.html atau pasien.html ini tersembunyi, walaupun user menuju ke web /admin.html ataupun sebaliknya misal admin menuju ke web /patient.html maka sistem atau website akan mendeteksi perilaku tersebut dan langsung melempar ke /dashboard masing masing user ataupun admin atau bisa juga menuju halaman login jika user belum login/register. Sebenarnya hal ini bisa dicegah dengan menggunakan encoding atau dilakukan oleh sisi server (seperti routing) tapi ini tidak memungkinkan bug atau mengalami kebocoran informasi. Hal seperti ini dilakukan agar mudah di praktikan saja, akan tetapi keamanan tetap bagus dan stabil.
4	1	Kenapa saat kita memasukan form dokter atau admin tidak mau menambahkan dokter dan muncul di console log itu forbidden	Karena aplikasi pada project yang dibuat memiliki keamanan yang dimana jika user tidak melakukan aktifitas atau melakukan aktifitas namun melewati batas yang di tentukan (1-15 menit) maka akan otomatis keluar dan jika di refresh akan menampilkan json seperti ini <pre>{   error:"unauthorized" }</pre>

			Yang dimana error tersebut mengatakan "tidak diizinkan" dan admin ataupun user wajib melakukan login ulang kembali. Guna mencegah mendeteksi fraud, dan juga diberikan rate limiting pada login, register, dashboard admin/dokter & dashboard user/pasien.
5	1	Darimana kita mendapatkan feedback setelah mengisi keluhan	Feedback yang di dapat itu berasal dari WhatsApp, akan tetapi setelah kirim keluhan, pasien menunggu dokter menjawab pesan WhatsApp yang nanti akan dikirimkan dari dokter terus ke bot lalu diteruskan ke pasien.
6	2	kenapa Ketika admin menambah data dokter baru, statusnya berhasil ditambah akan tetapi datanya tidak muncul di admin ?	Itu merupakan fake alert, bertujuan agar admin tidak terlalu lama lama untuk menginput data dosen yang bisa terbilang cukup simple, yang dimana hanya ada pengisian form Nama Dokter, Spesialisasinya, Nomor Whatsapp, dan Foto Dokter (menggunakan url website atau disimpan di server lalu dipanggil direktori folder gambarnya yaitu di <a href="http://localhost:3000/images/">http://localhost:3000/images/</a> yang dimana akan menampilkan json link url tersebut dan akan menjadi seperti ini <a href="http://localhost:3000/gambar/alya.jpg">http://localhost:3000/gambar/alya.jpg</a> dan folder gambar maupun images tidak memiliki kerentanan direktori listing yang dimana bug tersebut menampilkan daftar semua file ketika tidak ada file indeks, seperti index.php, index.html dan default.asp dalam direktori situs web tertentu. seperti 'ls' pada sistem Unix dan Linux dan 'dir' pada Windows. Yang dimana bug tersebut akan menjadi membocorkan informasi.) dan ketika di refresh akan muncul authorized yang dimana user ataupun admin harus melakukan aktifitas login kembali guna merefresh token yang telah di generate server, dan setelah login kembali, jika admin ingin menambah dokter baru maka status akan berhasil dan data akan terlihat, maka pada project ini, refresh sangat diperlukan.

## 4.4 Perbaikan Hasil Uji

Tabel 4 16 Perbaikan Hasil Uji

No	Pertanyaan	Jawaban Lengkap
1	Jika user atau pengguna lupa kata sandi itu bagaimana?	<b>Sistem menyediakan dua metode:</b> 1. <b>Pemulihan Mandiri dengan OTP:</b> - Masuk ke halaman "Lupa Password". - Masukkan nomor telepon terdaftar. - Sistem kirim OTP 6 digit via WhatsApp. - OTP berlaku terbatas waktu (misalnya 2 menit). - Setelah verifikasi OTP, pengguna bisa atur ulang kata sandi. 2. <b>Bantuan dari Admin:</b> - Jika gagal dengan OTP, pengguna bisa hubungi admin (misalnya via Instagram) untuk bantuan manual.
2	Feedback atau umpan balik dari dokter tidak terlihat di user?	Betul, <b>feedback dari dokter tidak ditampilkan di dashboard</b> , melainkan dikirim langsung ke WhatsApp pasien. Alasan: - <b>Privasi &amp; Kecepatan:</b> WhatsApp lebih cepat dan pribadi. - <b>Alur Sistem:</b> Website hanya untuk input keluhan dan login, sedangkan WhatsApp untuk konsultasi. - <b>Teknis:</b> Real-time chat di web butuh WebSocket, dan belum diterapkan. Namun, fitur sinkronisasi riwayat akan jadi prioritas pengembangan selanjutnya.
3	Kenapa menggunakan admin.html?	Penggunaan <b>admin.html &amp; patient.html</b> adalah untuk membedakan tampilan pengguna berdasarkan peran. Namun, <b>keamanan dijaga di sisi backend</b> , yaitu: - <b>Role-Based Routing:</b> Semua akses melalui /dashboard, lalu dicek req.session.user.role untuk menentukan tampilan. - <b>Middleware Otorisasi:</b> Ada middleware isAuthenticated dan isAdmin. Jika pasien akses /admin.html langsung, sistem akan menolak dan arahkan ulang ke halaman yang sesuai. Jadi, tidak bisa akses langsung tanpa otorisasi.
4	Kenapa saat memasukkan form dokter/admin muncul error "forbidden"?	Error <b>403 Forbidden</b> muncul karena dua hal: 1. <b>Session Timeout:</b> Sesi login habis karena tidak aktif (misalnya 15 menit). Solusi: login ulang. 2. <b>Token CSRF Tidak Valid:</b> Token CSRF punya masa berlaku. Jika halaman terlalu lama terbuka, token bisa kedaluwarsa. Solusi: refresh halaman atau login ulang sebelum kirim form.
5	Darimana kita mendapatkan feedback setelah mengisi keluhan?	Setelah kirim keluhan di website: - Sistem backend simpan keluhan. - Bot WhatsApp (bot.js) kirim pesan keluhan ke dokter. - Dokter membalas via WhatsApp. - Bot menangkap balasan dan kirimkan langsung ke WhatsApp Anda. <b>Kesimpulan:</b> Balasan dari dokter hanya dikirim melalui WhatsApp, bukan ditampilkan di web.

### 1. Jika user atau pengguna lupa kata sandi itu bagaimana?

Index.js

```
938  async function sendOtpViaBot(phone, otp) {
939    try {
940      const message = `Kode OTP Anda adalah ${otp}. Berlaku selama 2 menit.`;
941      const targetJid = `${phone}@s.whatsapp.net`;
942
943      await sock.sendMessage(targetJid, { text: message });
944      console.log(`Kode OTP berhasil dikirim ke nomor ${phone}`);
945      return true; // Indikasi sukses
946    } catch (error) {
947      console.error(`Gagal mengirim kode OTP ke nomor ${phone}:`, error);
948      return false; // Indikasi gagal
949    }
950  }
951
```

Gambar 4.6 Index.js

Keamanan agar tidak bisa di spam OTP

```
89  const otpLimiter = rateLimit({
90    windowMs: 10 * 60 * 1000, // 10 menit
91    max: 20, // Maksimum 5 permintaan OTP
92    message: "Terlalu banyak permintaan OTP, coba lagi nanti.",
93  });
94
```

Gambar 4.7 Keamanan agar tidak bisa di spam OTP

Validasi, Keamanan yang diterapkan pada bagian kode OTP dan Perubahan Password

```
541 app.post('/forgot-password/send-otp', [  
542   body('phone')  
543     .isMobilePhone('any', { strictMode: false }) // Memastikan nomor telepon valid  
544     .withMessage('Nomor telepon tidak valid'),  
545   body('phone')  
546     .notEmpty()  
547     .withMessage('Nomor telepon tidak boleh kosong'),  
548   body('phone')  
549     .isLength({ max: 15 })  
550     .withMessage('Nomor telepon tidak boleh lebih dari 15 karakter'),  
551 ], otpLimiter, async (req, res) => {  
552   const errors = validationResult(req);  
553   if (!errors.isEmpty()) {  
554     return res.status(400).json({ errors: errors.array() });  
555   }  
556  
557   const { phone } = req.body;  
558  
559   // Validasi nomor telepon  
560   if (!req.body.phone || typeof req.body.phone !== 'string') {  
561     return res.status(400).json({ error: 'Nomor telepon tidak valid' });  
562   }  
563  
564   // Periksa apakah nomor telepon terdaftar  
565   db.get('SELECT * FROM users WHERE username = ?', [phone], async (err, user) => {  
566     if (err) {  
567       console.error('Database error:', err);  
568       return res.status(500).json({ error: 'Terjadi kesalahan pada server.' });  
569     }  
570  
571     if (!user) {  
572       return res.status(404).json({ error: 'Nomor tersebut tidak tersedia.' });  
573     }  
574  
575     // Periksa apakah nomor sudah mencapai batas permintaan  
576     const now = Date.now();  
577     if (!otpRequestStore[phone]) {  
578       otpRequestStore[phone] = { count: 0, lastRequestTime: 0 };
```

Gambar 4.8 kode OTP dan Perubahan Password

```

579     }
580
581     const { count, lastRequestTime } = otpRequestStore[phone];
582
583     // Jika sudah mencapai batas 3 permintaan, periksa waktu tunggu
584     if (count >= 3) {
585         // const waitTime = Math.random() * (10 - 1) + 1; // Random antara 1 hingga 10 menit
586         const waitTime = Math.random() * (2 - 1) + 1; // Random antara 1 hingga 2 menit
587         const waitUntil = lastRequestTime + waitTime * 60 * 1000; // Waktu tunggu dalam milidetik
588
589         if (now < waitUntil) {
590             const remainingTime = Math.ceil((waitUntil - now) / 1000 / 60); // Sisa waktu dalam menit
591             return res.status(429).json({ error: 'Anda telah mencapai batas permintaan OTP. Silakan coba lagi dalam ${remainingTime} menit.' });
592         }
593
594         // Reset hitungan setelah waktu tunggu selesai
595         otpRequestStore[phone].count = 0;
596     }
597
598     // Generate OTP 6 digit
599     const otp = crypto.randomInt(100000, 999999);
600     otpStore[phone] = { otp, expiresAt: now + 2 * 60 * 1000 }; // OTP berlaku selama 2 menit
601
602     // Kirim OTP melalui bot WhatsApp
603     const success = await sendOtpViaBot(phone, otp);
604     if (success) {
605         otpRequestStore[phone].count += 1; // Tambahkan hitungan permintaan
606         otpRequestStore[phone].lastRequestTime = now; // Perbarui waktu permintaan terakhir
607         res.json({ status: 'success', message: 'Kode OTP telah dikirim.' });
608     } else {
609         res.status(500).json({ error: 'Gagal mengirim kode OTP.' });
610     }
611     });
612 });
613
614 app.post('/forgot-password/verify-otp', otpLimiter, [
615     body('phone')
616     .isMobilePhone('any', { strictMode: false }) // Memastikan nomor telepon valid
617     .withMessage('Nomor telepon tidak valid').notEmpty().withMessage('Nomor telepon tidak boleh kosong')
618     .isLength({ max: 15 }).withMessage('Nomor telepon tidak boleh lebih dari 15 karakter'),
619     body('otp')
620     .isNumeric().withMessage('Kode OTP harus berupa angka')
621     .isLength({ min: 6, max: 6 }).withMessage('Kode OTP harus terdiri dari 6 digit'),

```

Gambar 4.9

```

622 }, (req, res) => {
623   if (!errors.isEmpty()) {
624     return res.status(400).json({ errors: errors.array() });
625   }
626
627   const { phone, otp } = req.body;
628
629   // Validasi nomor telepon dan OTP
630   if (!req.body.phone || typeof req.body.phone !== 'string') {
631     return res.status(400).json({ error: 'Nomor telepon tidak valid' });
632   }
633
634   if (!req.body.otp || typeof req.body.otp !== 'string' || isNaN(req.body.otp)) {
635     return res.status(400).json({ error: 'Kode OTP tidak valid' });
636   }
637
638   const storedOtp = otpStore[phone];
639   if (!storedOtp) {
640     return res.status(400).json({ error: 'Kode OTP tidak ditemukan atau telah kedaluwarsa.' });
641   }
642
643   if (storedOtp.otp !== parseInt(otp)) {
644     return res.status(400).json({ error: 'Kode OTP tidak valid.' });
645   }
646
647   if (Date.now() > storedOtp.expiresAt) {
648     delete otpStore[phone];
649     return res.status(400).json({ error: 'Kode OTP telah kedaluwarsa.' });
650   }
651
652   // OTP valid, hapus dari store
653   delete otpStore[phone];
654   res.json({ status: 'success', message: 'Kode OTP valid. Anda dapat mengubah password.' });
655
656   });
657
658   app.post('/forgot-password/change-password', [
659     body('phone')
660       .isMobilePhone('any', { strictMode: false }) // Memastikan nomor telepon valid
661       .withMessage('Nomor telepon tidak valid').notEmpty().withMessage('Nomor telepon tidak boleh kosong')
662       .isLength({ max: 15 }).withMessage('Nomor telepon tidak boleh lebih dari 15 karakter'),
663     body('newPassword')
664       .isLength({ min: 8 }).withMessage('Password harus memiliki minimal 8 karakter')
665       .not().matches(/\s/).withMessage('Password tidak boleh mengandung spasi'),
666     body('newPassword')
667       .isLength({ max: 30 }).withMessage('Password tidak boleh lebih dari 30 karakter'),

```

Gambar 4.10



```

667 // isLength({ min: 8 }); withMessage('Password tidak boleh lebih dari 8 karakter ');
668 body('newPassword')
669   .notEmpty().withMessage('Password baru tidak boleh kosong'),
670   (req, res) => {
671     const errors = validationResult(req);
672     if (!errors.isEmpty()) {
673       return res.status(400).json({ errors: errors.array() });
674     }
675
676     const { phone, newPassword } = req.body;
677
678     // Validasi nomor telepon dan password baru
679     if (!req.body.phone || typeof req.body.phone !== 'string') {
680       return res.status(400).json({ error: 'Nomor telepon tidak valid' });
681     }
682     if (!req.body.newPassword || typeof req.body.newPassword !== 'string') {
683       return res.status(400).json({ error: 'Password baru tidak valid' });
684     }
685     if (/\/s/.test(req.body.newPassword)) {
686       return res.status(400).json({ error: 'Password baru tidak boleh mengandung spasi' });
687     }
688     if (req.body.newPassword.length < 8) {
689       return res.status(400).json({ error: 'Password baru harus memiliki minimal 8 karakter' });
690     }
691     if (req.body.newPassword.length > 30) {
692       return res.status(400).json({ error: 'Password baru tidak boleh lebih dari 30 karakter' });
693     }
694     if (!req.body.newPassword) {
695       return res.status(400).json({ error: 'Password baru tidak boleh kosong' });
696     }
697
698     // Perbarui password di database
699     db.run('UPDATE users SET password = ? WHERE username = ?', [newPassword, phone], (err) => {
700       if (err) {
701         console.error('Database error:', err);
702         return res.status(500).json({ error: 'Gagal mengubah password.' });
703       }
704       any
705       res.json({ status: 'success', message: 'Password berhasil diubah.' });
706     });
707   });
708

```

Gambar 4.11

## 2. Feedback atau umpan balik dari dokter tidak terlihat di user?

bot.js

Sistem/Logika meneruskan pesan dari Pasien ke Dokter dan Dokter ke Pasien

```
// Event utama yang menangani pesan masuk
sock.ev.on("messages.upsert", async ({ messages }) => {
  const msg = messages[0];
  if (!msg.message || msg.key.fromMe) return;

  const sender = msg.key.remoteJid;
  const senderNumber = sender.split('@')[0];
  const messageContent = msg.message.conversation || msg.message.extendedTextMessage?.text || "";

  // Cari sesi konsultasi yang aktif berdasarkan nomor pengirim (bisa pasien atau dokter)
  db.get(
    `SELECT b.id AS booking_id, u.username AS patient_phone, d.phone AS doctor_phone, b.is_active
    FROM bookings b
    JOIN users u ON b.user_id = u.id
    JOIN doctors d ON b.doctor_id = d.id
    WHERE (u.username = ? OR d.phone = ?) AND b.is_active = 1
    ORDER BY b.created_at DESC LIMIT 1`,
    [senderNumber, senderNumber],
    async (err, booking) => {
      if (err || !booking) return; // Abaikan jika tidak ada sesi aktif

      const pasienJid = `${booking.patient_phone}@s.whatsapp.net`;
      const dokterJid = `${booking.doctor_phone}@s.whatsapp.net`;
      let targetJid;
      let fromRole;

      // Tentukan siapa pengirim dan siapa penerima
      if (sender === dokterJid) {
        targetJid = pasienJid;
        fromRole = "doctor";
      } else if (sender === pasienJid) {
        targetJid = dokterJid;
        fromRole = "patient";
      } else {
        return;
      }

      // Teruskan pesan ke target
      await sock.sendMessage(targetJid, { text: messageContent });

      // Simpan pesan ke database
      db.run(`INSERT INTO messages (booking_id, sender, message) VALUES (?, ?, ?)`,
        [booking.booking_id, fromRole, messageContent]);
    }
  );
});
```

Gambar 4.12 bot.js

### 3. Kenapa menggunakan admin.html?

Index.js

Melakukan routing

```
258 app.get('/dashboard', isAuthenticated, (req, res) => {
259   if (!req.session.user) return res.redirect('/');
260
261   if (req.session.user.role === 'admin') {
262     res.sendFile(path.join(__dirname, 'public/admin.html'));
263   } else {
264     res.sendFile(path.join(__dirname, 'public/patient.html'));
265   }
266 });
267
```

Gambar 4.13 index.js

Middleware otorisasi (Keamanan, diberikan mana akses untuk admin, mana untuk user)

```
96 function isAuthenticated(req, res, next) {
97   // if (req.session.user) {
98     if (req.session && req.session.user) {
99       return next();
100     }
101     res.status(403).json({ error: 'Unauthorized' });
102   }
103
104   function isAdmin(req, res, next) {
105     // if (req.session.user && req.session.user.role === 'admin') {
106       if (req.session && req.session.user && req.session.user.role === 'admin') {
107         return next();
108       }
109       res.status(403).json({ error: 'Unauthorized - Admins only' });
110     }
111
112     function isUser(req, res, next) {
113       // if (req.session.user && req.session.user.role === 'patient') {
114       if (req.session && req.session.user && req.session.user.role === 'patient') {
115         return next();
116       }
117       res.status(403).json({ error: 'Unauthorized - Users only' });
118     }
119   }
120 }
```

Gambar 4.14 Middleware otomatis

### 4. Kenapa saat memasukkan form dokter/admin muncul error "forbidden"?

Keamanan Sesi berakhir, CSRF Token agar tidak ada kerentanan CSRF Injection/Penginputan data secara tidak sah, beberapa Akses url yang diizinkan, dan Rate Limit agar tidak ada kerentanan DDoS

## Index.js

```
37 app.use(session({
38   secret: randomHex,
39   resave: false,
40   saveUninitialized: true,
41   cookie: {
42     secure: process.env.NODE_ENV === 'production',
43     // secure: process.env.NODE_ENV === 'production' ? true : false,
44     maxAge: 1000 * 60 * 60 * 24, // 1 day
45     sameSite: 'strict',
46     httpOnly: true,
47   }
48 }));
49
50 app.use(helmet());
51 app.use(helmet.contentSecurityPolicy({
52   directives: {
53     defaultSrc: ["'self'"],
54     scriptSrc: ["'self'", "unsafe-inline", "https://cdn.jsdelivr.net", "https://cdnjs.cloudflare.com"],
55     styleSrc: ["'self'", "unsafe-inline", "https://cdn.jsdelivr.net", "https://cdnjs.cloudflare.com"],
56     imgSrc: ["'self'", "*"],
57     // scriptSrcAttr: ["'none'"],
58     // styleSrcAttr: ["'none'"],
59     scriptSrcAttr: ["'self'", "unsafe-inline", "https://cdn.jsdelivr.net", "https://cdnjs.cloudflare.com"],
60     styleSrcAttr: ["'self'", "unsafe-inline", "https://cdn.jsdelivr.net", "https://cdnjs.cloudflare.com"],
61   }
62 }));
63
64 const csrfProtection = csrf({ cookie: true });
65 app.use(csrfProtection);
66
67 const limiter = ratelimit({
68   windowMs: 15 * 60 * 1000, // 15 menit
69   max: 50, // Maksimum 13 percakapan
70   message: "Terlalu banyak permintaan, coba lagi nanti. tunggu 15 menit.",
71   standardHeaders: true,
72   legacyHeaders: false,
73 });
74
75 app.use(limiter);
76
77 const loginLimiter = ratelimit({
78   windowMs: 15 * 60 * 1000, // 15 menit
```

Gambar 4.15 Index.js

```
79   max: 5, // Maksimum 5 permintaan login
80   message: "Terlalu banyak permintaan login, coba lagi nanti.",
81 });
82
83 const registerLimiter = ratelimit({
84   windowMs: 15 * 60 * 1000, // 15 menit
85   max: 5, // Maksimum 5 permintaan registrasi
86   message: "Terlalu banyak permintaan registrasi, coba lagi nanti.",
87 });
88
89 const otpLimiter = ratelimit({
90   windowMs: 10 * 60 * 1000, // 10 menit
91   max: 20, // Maksimum 5 permintaan OTP
92   message: "Terlalu banyak permintaan OTP, coba lagi nanti.",
93 });
94
```

Gambar 4.16

```

188 app.get('/csrf-token', (req, res) => {
189   res.json({ csrfToken: req.csrfToken() });
190 });
191
192 app.use((err, req, res, next) => {
193   if (err.code === 'EBADCSRFTOKEN') {
194     // Token CSRF tidak valid
195     res.status(403).json({ error: 'Invalid CSRF token' });
196   } else {
197     next(err);
198   }
199 });

```

Gambar 4. 17

Admin.html

```

721   fetch('/csrf-token')
722     .then(res => res.json())
723     .then(data => {
724       document.getElementById('csrfToken').value = data.csrfToken;
725     })
726     .catch(err => {
727       console.error('Gagal mengambil token CSRF:', err);
728       // alert('Gagal memuat token CSRF. Silakan muat ulang halaman.');
```

Gambar 4.18 Admin.html

```

583 <form id="addDoctorForm" novalidate autocomplete="on">
584   <input type="hidden" id="csrfToken" name="_csrf" value="">
585   <label for="name">Nama Dokter:</label>
586   <input type="text" id="name" name="name" pattern="[A-Za-z .,']+$" placeholder="Nama Dokter" required autocomplete="name">
587   <label for="specialization">Spesialisasi:</label>
588   <input type="text" id="specialization" name="specialization" pattern="[A-Za-z .,']+$" placeholder="Spesialisasi" required autocomplete="organization-title">
589   <label for="phone">Nomor WhatsApp:</label>
590   <input type="text" id="phone" name="phone" pattern="(62)[0-9]{8,13}" placeholder="Nomor WhatsApp (62xxx)" required autocomplete="tel">
591   <label for="photo_url">Foto Dokter (URL):</label>
592   <input type="text" id="photo_url" name="photo_url" placeholder="Foto URL" required autocomplete="url">
593   <button type="submit">Tambah Dokter</button>
594   <button type="reset">Bersihkan Formulir</button>
595 </form>

```

Gambar 4.19

## 5. Darimana kita mendapatkan feedback setelah mengisi keluhan?

Sama seperti nomor 2

```
// Event utama yang menangani pesan masuk
sock.ev.on("messages.upsert", async ({ messages }) => {
  const msg = messages[0];
  if (!msg.message || msg.key.fromMe) return;

  const sender = msg.key.remoteJid;
  const senderNumber = sender.split('@')[0];
  const messageContent = msg.message.conversation || msg.message.extendedTextMessage?.text || "";

  // Cari sesi konsultasi yang aktif berdasarkan nomor pengirim (bisa pasien atau dokter)
  db.get(`
    SELECT b.id AS booking_id, u.username AS patient_phone, d.phone AS doctor_phone, b.is_active
    FROM bookings b
    JOIN users u ON b.user_id = u.id
    JOIN doctors d ON b.doctor_id = d.id
    WHERE (u.username = ? OR d.phone = ?) AND b.is_active = 1
    ORDER BY b.created_at DESC LIMIT 1`,
    [senderNumber, senderNumber],
    async (err, booking) => {
      if (err || !booking) return; // Abaikan jika tidak ada sesi aktif

      const pasienJid = `${booking.patient_phone}@s.whatsapp.net`;
      const dokterJid = `${booking.doctor_phone}@s.whatsapp.net`;
      let targetJid;
      let fromRole;

      // Tentukan siapa pengirim dan siapa penerima
      if (sender === dokterJid) {
        targetJid = pasienJid;
        fromRole = "doctor";
      } else if (sender === pasienJid) {
        targetJid = dokterJid;
        fromRole = "patient";
      } else {
        return;
      }

      // Teruskan pesan ke target
      await sock.sendMessage(targetJid, { text: messageContent });

      // Simpan pesan ke database
      db.run(`INSERT INTO messages (booking_id, sender, message) VALUES (?, ?, ?)`,
        [booking.booking_id, fromRole, messageContent]);
    }
  );
});
```

Gambar 4. 20

### Tambahan:

#### 1. Pencegahan XSS dan Validasi Form

Setiap file yang ada di folder Public tambahkan Library DOMPurify agar tidak ada XSS pada DOM javascript

```

8 + <script src="https://cdnjs.cloudflare.com/ajax/libs/dompurify/3.2.5/purify.min.js"
    integrity="sha512-/CUTa84sWkqWlEBejNrrtW7Yc4ctH3Ome2ymvCKOo9YcZ4sh98tndUy4LutE2xGcAgD4fyz16y+gSyJdGCB5w==" crossorigin="anonymous"
    referrerpolicy="no-referrer"></script>

```

Gambar 4.21

Contoh Pencegahan DOMXSS dan validasi Inputan/Nomor:

```

4 + const username = document.getElementById('username').value.trim();
5 + const password = document.getElementById('password').value.trim();
6 +
7 + if (!username || !password) {
8 +   Swal.fire({
9 +     icon: 'error',
10 +     title: 'Oops...',
11 +     text: 'Username dan password harus diisi!',
12 +   });
13 +   return;
14 + }
15 +
16 + if (username.startsWith('0')) {
17 +   // username = username.replace('0', '62');
18 +   username = DOMPurify.sanitize(username.replace('0', '62'));
19 + } else if (username.startsWith('+62')) {
20 +   // username = username.replace('+62', '62');
21 +   username = DOMPurify.sanitize(username.replace('+62', '62'));
22 + }
23 +
24 + // Cegah XSS dengan meng-encode karakter berbahaya
25 + let sanitizedUsername = username.replace(/[<>"]'/g, '');
26 + let sanitizedPassword = password.replace(/[<>"]'/g, '');
27 +
28 + sanitizedUsername = DOMPurify.sanitize(sanitizedUsername);
29 + sanitizedPassword = DOMPurify.sanitize(sanitizedPassword);

```

Gambar 4.22

Dan juga contoh untuk minimum input:

```

186 + <input type="text" id="username" maxlength="15" placeholder="Username Ex: 62xxx" required>
187 + <input type="password" id="password" minlength="8" placeholder="Password" required>

```

Gambar 4.23

Dan begitupun semuanya yang ada di folder Public (semua file .html wajib dimasukan DOMPurify dan Validasi serta Minimum/Maximum Input)

## 2. Pencegahan Dirlisting / Directory Listing + Cookie management / Token Management (Menghindari CSRF Injection)

Pencegahan Direktory Listing itu penting, karena kerentanan ini berbahaya, seseorang yang tidak memiliki akses dapat masuk ke folder/direktori yang tidak sah

```
27 app.use(bodyParser.urlencoded({ extended: true }));
28 app.use(bodyParser.json());
29 app.use(cookieParser());
30
31 app.use('/gambar', express.static(path.join(__dirname, 'public/gambar')));
32
33 app.use(express.static('public'));
34 // app.use('/gambar', express.static(path.join(__dirname, 'public/gambar')));
35
```

Gambar 4.24

```
863 document.getElementById('logoutButton').addEventListener('click', function() {
864     const csrfToken = document.getElementById('csrfToken').value;
865     Swal.fire({
866         title: 'Keluar?',
867         text: 'Apakah Anda yakin ingin keluar?',
868         icon: 'warning',
869         showCancelButton: true,
870         confirmButtonText: 'Ya, Keluar',
871         cancelButtonText: 'Tidak',
872         reverseButtons: true
873     }).then((result) => {
874         if (result.isConfirmed) {
875             fetch('/logout', { method: 'POST', headers: { 'X-CSRF-Token': csrfToken, 'CSRF-Token': csrfToken } })
876                 .then(res => {
877                     if (!res.ok) throw new Error('Gagal logout');
878                     window.location.href = '/';
879                 })
880                 .catch(error => console.error('Error:', error));
881         }
882     });
883 });
884
```

Gambar 4.25



```

938 // Jika semua validasi lolos, kirim data
939 fetch('/doctors', {
940   method: 'POST',
941   headers: {
942     'Content-Type': 'application/json',
943     'X-CSRF-Token': csrfToken,
944     'CSRF-Token': csrfToken
945   },
946   body: JSON.stringify({ name, specialization, phone, photo_url })
947 })
948 .then(res => {
949   if (!res.ok) throw new Error('Gagal menambah dokter');
950   return res.json();
951 })
952 .then(data => {
953   Swal.fire({
954     icon: 'success',
955     title: 'Berhasil',
956     text: 'Dokter berhasil ditambahkan!'
957   });
958   loadDoctors();
959   this.reset();
960 })
961 .catch(error => {
962   console.error('Error:', error);
963   Swal.fire({
964     icon: 'error',
965     title: 'Gagal',
966     text: 'Gagal menambahkan dokter. Silakan coba lagi.'
967   });
968 });
969 });
970
971 function deleteUser(userId) {
972   const csrfToken = document.getElementById('csrfToken').value; // Ambil token CSRF
973
974   Swal.fire({
975     title: 'Hapus Pengguna?',
976     text: 'Apakah Anda yakin ingin menghapus pengguna ini? Tindakan ini tidak dapat dibatalkan.',
977     icon: 'warning',
978     showCancelButton: true
979   });

```

Gambar 4.26

```

978     showCancelButton: true,
979     confirmButtonText: 'Ya, Hapus',
980     cancelButtonText: 'Tidak',
981     reverseButtons: true
982   }).then((result) => {
983     if (result.isConfirmed) {
984       fetch(`/admin/users/${userId}`, {
985         method: 'DELETE',
986         headers: {
987           'Content-Type': 'application/json',
988           'X-CSRF-Token': csrfToken // Sertakan token CSRF di header
989         }
990       })
991       .then(res => {
992         if (!res.ok) throw new Error('Gagal menghapus pengguna');
993         Swal.fire({
994           icon: 'success',
995           title: 'Pengguna Dihapus',
996           text: 'Pengguna berhasil dihapus.',
997         }).then(() => {
998           location.reload(); // Muat ulang halaman setelah penghapusan
999         });
1000       })
1001       .catch(error => {
1002         console.error('Error:', error);
1003         Swal.fire({
1004           icon: 'error',
1005           title: 'Gagal Menghapus Pengguna',
1006           text: 'Terjadi kesalahan saat menghapus pengguna. Silakan coba lagi.',
1007         });
1008       });
1009     }
1010   });
1011 }
1012
1013 function deleteDoctor(id) {
1014   const csrfToken = document.getElementById('csrfToken').value;
1015   Swal.fire({
1016     title: 'Hapus Dokter?',
1017     text: 'Apakah Anda yakin ingin menghapus dokter ini?',

```

Gambar 4.27

Dan seterusnya

3. Validasi URL, Nama, Nomor dan tipe file (Semua validasi sama diterapkan ke semua file .html)

```
885 document.getElementById('addDoctorForm').addEventListener('submit', function(e) {
886     e.preventDefault();
887     const name = DOMPurify.sanitize(document.getElementById('name').value.trim());
888     const specialization = DOMPurify.sanitize(document.getElementById('specialization').value.trim());
889     const phone = DOMPurify.sanitize(document.getElementById('phone').value.trim());
890     const photo_url = DOMPurify.sanitize(document.getElementById('photo_url').value.trim());
891
892     const csrfToken = document.getElementById('csrfToken').value;
893
894     // Validasi nama (hanya huruf, spasi, dan beberapa karakter khusus)
895     const nameRegex = /^[A-Za-z ., '-]+$/;
896     if (!nameRegex.test(name)) {
897         Swal.fire({
898             icon: 'error',
899             title: 'Error',
900             text: 'Nama dokter tidak valid. Hanya boleh berisi huruf, spasi, dan tanda baca sederhana.'
901         });
902         return;
903     }
904
905     // Validasi spesialisasi (hanya huruf, spasi, dan beberapa karakter khusus)
```

Gambar 4.28

```
903 }
904
905 // Validasi spesialisasi (hanya huruf, spasi, dan beberapa karakter khusus)
906 const specializationRegex = /^[A-Za-z ., '-]+$/;
907 if (!specializationRegex.test(specialization)) {
908     Swal.fire({
909         icon: 'error',
910         title: 'Error',
911         text: 'Spesialisasi tidak valid. Hanya boleh berisi huruf, spasi, dan tanda baca sederhana.'
912     });
913     return;
914 }
915
916 // Validasi nomor telepon (format 62xxx)
917 const phoneRegex = /^(62)[0-9]{8,13}$/;
918 if (!phoneRegex.test(phone)) {
919     Swal.fire({
920         icon: 'error',
921         title: 'Error',
922         text: 'Nomor WhatsApp tidak valid. Gunakan format 62xxx.'
923     });
924     return;
925 }
926
927 // Validasi URL foto
928 const urlRegex = /^https?:\/\/.+.(jpg|jpeg|png|gif)$/i;
929 if (!urlRegex.test(photo_url)) {
930     Swal.fire({
931         icon: 'error',
932         title: 'Error',
933         text: 'URL foto tidak valid. Gunakan URL yang diakhiri dengan .jpg, .jpeg, .png, atau .gif.'
934     });
935     return;
936 }
937
938 // Jika semua validasi lolos, kirim data
939 fetch('/doctors', {
940     method: 'POST',
941     headers: {
942         'Content-Type': 'application/json',
```

Gambar 4.29

## Peningkatan dan Implementasi Keamanan Sistem

Sebagai respons terhadap kebutuhan akan aplikasi yang aman dan untuk memperkuat sistem berdasarkan standar industri, beberapa lapisan keamanan telah diimplementasikan secara proaktif.

### 1. Validasi dan Sanitasi Input untuk Mencegah XSS:

- **Tindakan:** Untuk mencegah serangan *Cross-Site Scripting* (XSS), di mana penyerang bisa menyisipkan skrip berbahaya, sistem menerapkan validasi di dua lapisan.
- **Frontend:** *Library DOMPurify* digunakan di semua halaman formulir (register.html, admin.html, dll.) untuk membersihkan input dari pengguna sebelum dikirim ke server.
- **Backend:** *Middleware express-validator* digunakan untuk memvalidasi ulang semua data yang masuk di sisi server, memastikan format dan tipe data sesuai dengan yang diharapkan.

### 2. Perlindungan Terhadap Serangan CSRF (Cross-Site Request Forgery):

- **Tindakan:** Untuk mencegah penyerang melakukan aksi atas nama pengguna yang sudah login, *middleware csrf* diimplementasikan.
- **Mekanisme:** Sistem menghasilkan token CSRF yang unik untuk setiap sesi. Token ini wajib disertakan dalam setiap permintaan yang mengubah data (seperti POST atau DELETE). Server akan menolak permintaan apapun yang tidak memiliki token yang valid, seperti yang terlihat pada penanganan *error* EBADCSRFTOKEN.

### 3. Manajemen Sesi yang Aman:

- **Tindakan:** Konfigurasi sesi di `index.js` telah diperkuat untuk meningkatkan keamanan.
- **Mekanisme:**
  - **Session Timeout:** Sesi diatur untuk berakhir secara otomatis setelah **15 menit** tidak aktif (`maxAge: 15 * 60 * 1000`) untuk mencegah penyalahgunaan akun yang ditinggalkan terbuka.

- **Cookie httpOnly:** *Cookie* sesi diatur sebagai httpOnly, yang berarti *cookie* tersebut tidak dapat diakses melalui JavaScript di sisi klien, melindunginya dari pencurian oleh skrip XSS.

#### 4. Pencegahan Serangan Brute-Force dengan Rate Limiting:

- **Tindakan:** Untuk melindungi *endpoint* vital dari serangan *brute-force* (percobaan berulang-ulang), diterapkan pembatasan laju permintaan menggunakan **express-rate-limit**.
- **Mekanisme:** Batasan diterapkan pada:
  - **Login & Registrasi:** Maksimal 5 permintaan setiap 15 menit.
  - **Permintaan OTP:** Maksimal 20 permintaan setiap 10 menit untuk mencegah *spam* OTP ke nomor pengguna.

#### 5. Pengamanan Header HTTP:

- **Tindakan:** *Middleware helmet.js* digunakan untuk mengatur berbagai *header* HTTP yang aman secara otomatis.
- **Manfaat:** Ini membantu melindungi aplikasi dari serangan umum berbasis web seperti *clickjacking*, XSS, dan lainnya dengan menerapkan praktik keamanan standar industri.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Proyek aplikasi website Medical Conversation berhasil dikembangkan sebagai solusi komunikasi daring antara pasien dan tenaga medis. Aplikasi ini memudahkan konsultasi kesehatan secara efisien, cepat, dan aman melalui fitur percakapan teks, pencatatan riwayat konsultasi, serta antarmuka yang ramah pengguna. Dengan sistem ini, hambatan jarak dan waktu dapat diminimalkan. Proyek ini menjadi langkah awal menuju digitalisasi layanan kesehatan yang lebih praktis dan terjangkau. Dengan hadirnya Medical Conversation, inovasi ini diharapkan mampu mendukung transformasi digital dalam dunia kesehatan, memberikan akses layanan yang lebih inklusif, anonimitas dan nyaman bagi masyarakat, serta menjadi fondasi bagi pengembangan solusi medis berbasis teknologi di masa depan.

Berikut Role-Role yang diberikan pada project ini

1. Indra Dwi Aryadi (**Programmer/Fullstack Engineer, Cloud engineer, Security Advisor, Penetration Testing/Tester, Desain, Project Manager, Dan lain lain**)
2. Lingga Safitri (**Project Manager, Tester**)
3. Prayoga Pratama (**Project Manager, Advisor/Penasihat/Perekomendasian**)
4. Muhammad Rizki Ramadhan (**Project Manager, Tester**)

#### **5.2 Saran Dan Pengembangan Lanjutan**

Meskipun aplikasi ini telah berhasil dikembangkan sesuai dengan ruang lingkup yang ditentukan, masih terdapat banyak peluang untuk pengembangan lebih lanjut di masa mendatang. Beberapa saran untuk pengembangan selanjutnya adalah:

1. **Mengembangkan Aplikasi Mobile:** Membuat versi aplikasi untuk platform *mobile* (Android dan iOS) agar dapat menjangkau lebih banyak pengguna dan memberikan kemudahan akses melalui *smartphone*.
2. **Fitur Panggilan Video dan Suara:** Mengintegrasikan fitur panggilan video atau suara secara langsung di dalam aplikasi untuk memungkinkan sesi konsultasi yang lebih interaktif dan mendalam, tanpa harus bergantung pada aplikasi pihak ketiga.
3. **Manajemen Jadwal Dokter:** Menambahkan fitur bagi dokter untuk dapat mengatur jadwal ketersediaan mereka, sehingga pasien dapat memesan sesi konsultasi pada waktu yang pasti.
4. **Integrasi Sistem Pembayaran:** Mengimplementasikan *payment gateway* untuk memfasilitasi sesi konsultasi berbayar. Ini akan membuka model bisnis baru dan memungkinkan aplikasi digunakan secara komersial oleh klinik atau rumah sakit.
5. **Notifikasi Real-time di Aplikasi Web:** Menambahkan sistem notifikasi *push* pada *dashboard* admin dan pasien untuk memberikan pembaruan secara langsung (misalnya, saat ada pesan baru dari dokter) tanpa perlu *me-refresh* halaman.
6. **Peningkatan Fitur AI:** Mengembangkan kemampuan asisten AI pada bot WhatsApp agar tidak hanya menjawab pertanyaan umum, tetapi juga dapat membantu melakukan triase awal berdasarkan gejala yang dimasukkan oleh pasien.
7. **Enkripsi Pesan di Database:** Untuk meningkatkan keamanan dan privasi, menerapkan enkripsi pada data percakapan yang tersimpan di dalam tabel *messages* di *database*.

### 5.3 Keterbatasan Sistem

Sistem "Medical Conversation" yang telah dibangun memiliki beberapa keterbatasan yang perlu menjadi perhatian, di antaranya:

1. **Bukan Alat Diagnosis:** Aplikasi ini hanya berfungsi sebagai media komunikasi awal dan tidak dirancang untuk menggantikan proses diagnosis medis yang harus dilakukan oleh tenaga medis profesional secara langsung atau tatap muka.
2. **Ketergantungan Penuh pada WhatsApp:** Alur komunikasi inti dari sistem ini sangat bergantung pada fungsionalitas bot WhatsApp yang dibangun menggunakan *library* Baileys. Adanya gangguan pada layanan WhatsApp, perubahan kebijakan API, atau masalah pada *library* dapat secara langsung mengganggu jalannya proses konsultasi.
3. **Manajemen Sesi yang Sederhana:** Manajemen sesi konsultasi oleh dokter saat menangani beberapa pasien sekaligus masih bersifat manual dan bergantung pada perintah-perintah teks (seperti `.listpasien` dan `. kirim`) melalui WhatsApp.
4. **Sesi Login dengan Batas Waktu:** Demi keamanan, sesi login pengguna (baik admin maupun pasien) memiliki batas waktu. Jika tidak ada aktivitas dalam rentang waktu tertentu, sistem akan secara otomatis mengeluarkan pengguna, yang kemudian harus melakukan login ulang untuk dapat mengakses sistem kembali.



# LAMPIRAN

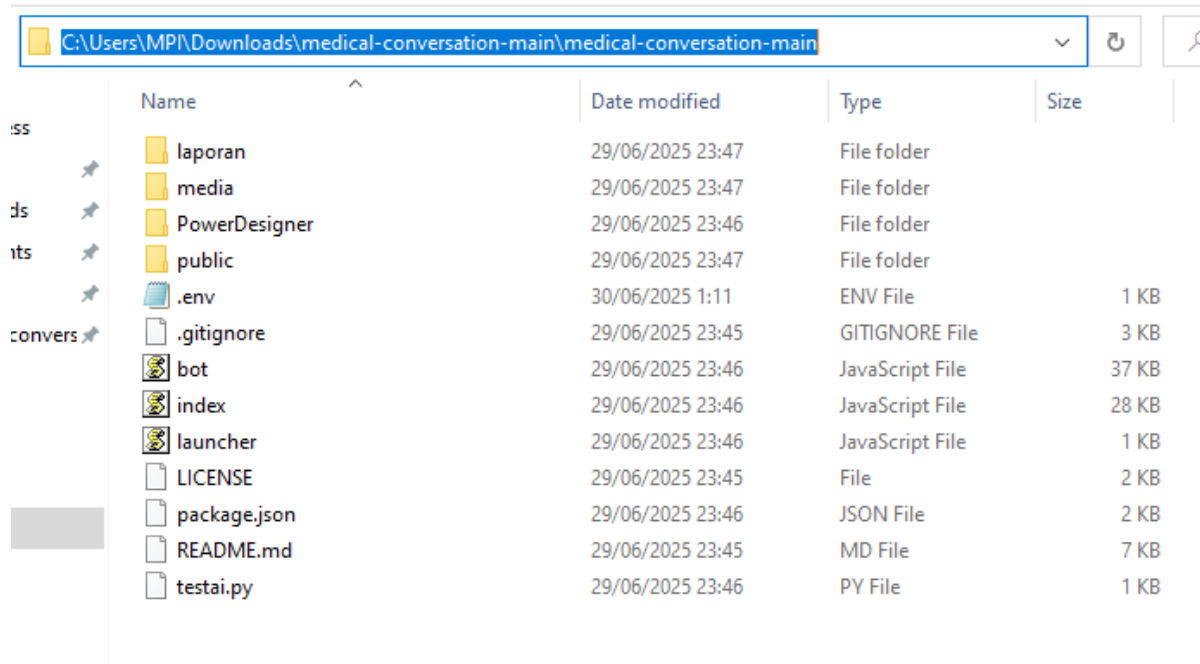
## INSTALASI DAN PENGUNAAN/KONFIGURASI

- Pertama tama kita harus menyalakan internet terlebih dahulu
- Buka browser dan klik <https://github.com/Xnuvers007/medical-conversation/archive/refs/heads/main.zip>
  - Setelah itu, kita download aplikasi atau software ekstrak seperti 7ZIP atau WinRar, lakukan instalasi software seperti biasa (jika belum mempunyai aplikasi pengekstrak file) tapi biasanya di windows 10 keatas itu sudah ada secara otomatis (Developernya pakai windows 10 hehe #Indra)
- Setelah itu maka akan terdapat file zip yang bernama “medical-conversation-main.zip”, silakan di ekstrak aplikasinya, disini saya merekomendasikan menggunakan WinRar, di winrar, tapi jika sudah paham dengan software aplikasi zip lainnya, silakan. Pada panduan ini, kita menggunakan bawaan dari windows
- Buka aplikasi tersebut dengan cara klik kanan lalu pilih “extract all” lalu pilih saja langsung “extract”, kalau di winrar, kalian bisa klik kanan pada file tersebut lalu pilih “Ekstrak disini” atau “Extract here”
- Lalu masuk ke dalam folder “medical-conversation-main”
- Setelah terbuka, kita membutuhkan framework NodeJS, yang dimana kita dapat menginstall nya ke halaman situs resmi nodejs nya yaitu: <https://nodejs.org/id> yang dimana kita dapat mengunduhnya di halaman unduh (<https://nodejs.org/id/download>) kita harus menginstall NodeJS versi 20 keatas disarankan menggunakan Versi paling terbaru saat ini (30/06/2025) yaitu v24.3.0 silakan gulir ke bawah halaman situsnya, untuk sistem operasi di sesuaikan, jika windows maka pilih windows dan sesuaikan arsitektur nya (disini *architecture* saya adalah x64) lalu disesuaikan, pilih MSI atau standalone, disini yang mudah adalah MSI / MicroSoft Installer, jika Anda malas, Anda bisa gunakan link ini dan akan otomatis terdownload (<https://nodejs.org/dist/v24.3.0/node-v24.3.0-x64.msi>) silakan download

- Jika sudah selesai mendownload NodeJS yang MSI maka buka dan lakukan instalasi seperti biasa. klik kanan pilih install, lalu next, centang “I accept the terms in the License Agreement” lalu next, silakan masukan file instalasi nya ke folder yang Anda ingin pindahkan dengan cara memilih change, namun jika Anda ingin default “C:\Program Files\nodejs\” maka tidak apa apa dan bisa lanjut ke next. Disini saya menyarankan default saja lalu next. Jika sudah sampai custom setup, bisa langsung next saja, dan lanjut next lagi, dan pilih install. Jika muncul “User Account Control” pilih saja Yes dan tunggu hingga selesai. Lalu tekan tombol finish atau selesai.
- Untuk mengetes apakah NodeJS sudah terinstall atau belum, bisa seperti ini
  - Tekan Windows + R
  - Ketik cmd
  - Jika cmd sudah muncul ketikan 2 perintah ini
    - node -v
    - npm -v
- cari dan buka kembali folder “medical-conversation-main”
- klik kanan pada file “.env”
- pilih open with, setelah itu buka “more apps” lalu pilih aplikasi notepad. Lalu atur sendiri password dan username serta node\_env nya (bisa development atau production). Setelah itu pada variabel “DEEPSEEKER\_API\_KEY=” kita harus ke browser terlebih dahulu ke link berikut: <https://openrouter.ai/deepseek/deepseek-prover-v2:free/api> Silakan pilih “Sign In” jika belum punya akun silakan Sign Up (setelah tekan tombol sign in) silakan daftar masing masing dengan akun dan password yang kalian gunakan.
  - Jika kalian tidak punya akun, kalian bisa gunakan akun yang saya kasih yaitu akun github. Dan login saja <https://github.com/login>
    - Username/Email: Zeronyth atau [socalsamiam@peakwavepro.com](mailto:socalsamiam@peakwavepro.com)
    - Password: Acumalaka\_123
  - Setelah login dari github, Anda bisa balik menuju <https://openrouter.ai/deepseek/deepseek-prover-v2:free/api> dan silakan login dengan github
  - Jika tidak bisa menggunakan Github, gunakan email dan password ini (Sign In)

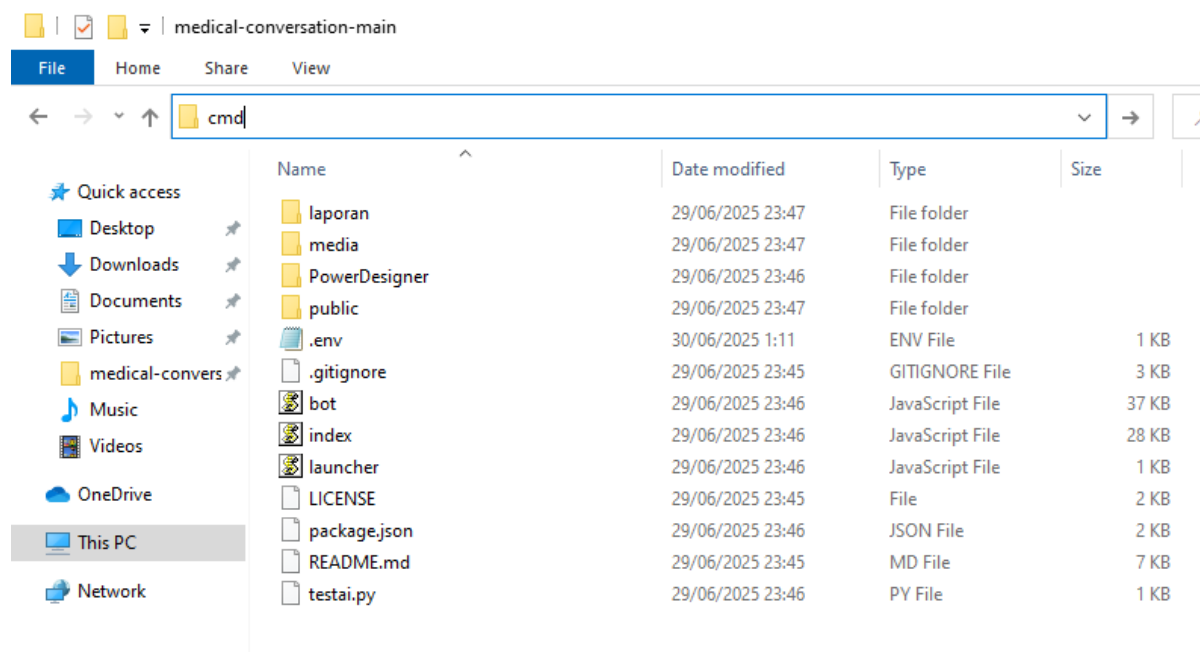
- Email: [c.riz.a.noi.ezi.n@gmail.com](mailto:c.riz.a.noi.ezi.n@gmail.com)
  - Password: Acumalaka\_123
- Jika tidak bisa, gunakan email masing masing yang masih bisa kalian akses kendali penuh dan jangan lupa untuk verifikasi Akun
- Jika kalian diarahkan ke halaman lain, langsung saja masuk ke: <https://openrouter.ai/deepseek/deepseek-prover-v2:free/api> lalu gulir kebawah sampai Anda menemukan tombol “Create API Key” maka Anda akan diarahkan ke sini <https://openrouter.ai/settings/keys> setelah itu anda menekan tombol “Create API Key” lagi
- Masukan saja Nama dan Limit, kosongkan limit jika ingin tanpa batas kreditnya, disini saya menamakan “WhatsappMPI” dan mengosongkan limit. Lalu setelah itu tekan tombol “Create” dan salin API Key tersebut dan jangan sampai hilang!
  - Apikey: `sk-or-v1-2e32ce1fa40d8d5534c2b6f48bd770f2294a22fd954f64377ada0a791ddf0`
- Lalu masukan API Key tersebut ke dalam file “.env” tersebut dan isi pada bagian `DEEPSEEKER_API_KEY=` dengan ini `sk-or-v1-2e32ce1fa40d8d5534c2b6f48bd770f2294a22fd954f64377ada0a791ddf0`
  - `DEEPSEEKER_API_KEY=` `sk-or-v1-2e32ce1fa40d8d5534c2b6f48bd770f2294a22fd954f64377ada0a791ddf0`
- Lalu save
- Setelah cek apakah ada git atau tidak, cara ceknya adalah tekan windows + r (tekan windows dibarengi dengan r) lalu ketikan “git -v” jika tidak ada maka langkah selanjutnya yaitu install git, kalian bisa menggunakan link ini untuk menuju ke git (<https://git-scm.com/downloads>) dan sesuaikan sistem operasi yang Anda gunakan, disini saya menggunakan windows, maka saya download yang windows dan juga sesuaikan architecturenya, apakah menggunakan x64 atau x86 atau ARM64 dan lain lain
- Anda bisa menekan link pada tulisan “Click here to download” atau “Git for Windows/x64 Setup” jika Anda menggunakan windows dengan arsitektur 64 bit
  - Jika Anda malas, maka gunakan link berikut: <https://github.com/git-for-windows/git/releases/download/v2.50.0.windows.1/Git-2.50.0-64-bit.exe>

- Setelah itu lakukan penginstalan aplikasi Git seperti biasanya, atau kalau mau lebih cepat adalah tinggal next next terus saja hingga selesai penginstalan
  - Buka aplikasi git yang sudah di download
  - Double click atau klik dua kali
  - Pilih yes jika user account control nya muncul
  - Pilih next
  - Lokasi arahkan ke C:\Program Files\Git , jika sudah maka next
  - Additional Icons bisa di centang bisa tidak, selebihnya tidak usah diapa apakan, klik lanjut
  - Next lagi, biarkan menjadi “Git”
  - Jika muncul “use vim” tinggal next saja
  - Pilih let git decide -> next
  - Pilih yang ke 2 “git from the command line.....” -> next
  - Pilih “use bundled openssh” -> next
  - Pilih “use the native windows secure channel.....” -> next
  - Pilih yang pertama “checkout the windows style....” -> next
  - Pilih “Use MinTTY” -> next
  - Pilih “fast forward or merge” -> next
  - Pilih “git credential manager” -> next
  - Ceklis “enable file system caching” -> install
  - Silakan menunggu hingga berhasil
  - Jika sudah selesai, jangan ceklis ke 2 nya, lalu finish
- Setelah itu, buka folder “medical-conversation-main”. Pada bagian lokasi folder tersebut diubah menjadi cmd, contoh:  
 Dari seperti ini



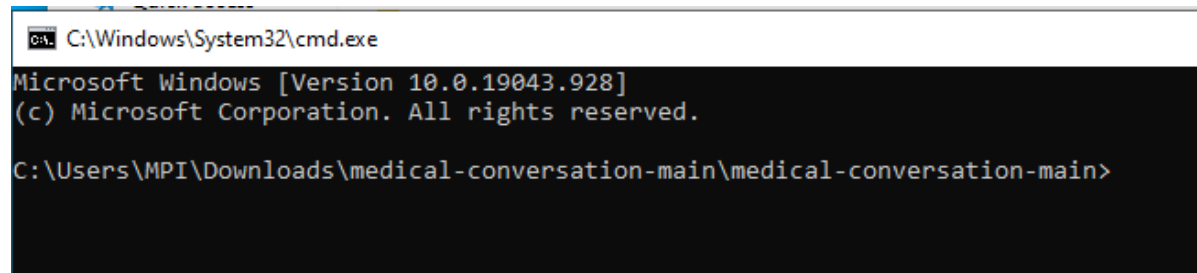
Gambar 29 C:\User\MPI\Downloads\medical-convesation

Menjadi seperti ini



Gambar 1 This PC

Maka cmd akan keluar



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\MPI\Downloads\medical-conversation-main\medical-conversation-main>
```

*Gambar 2 cmd*

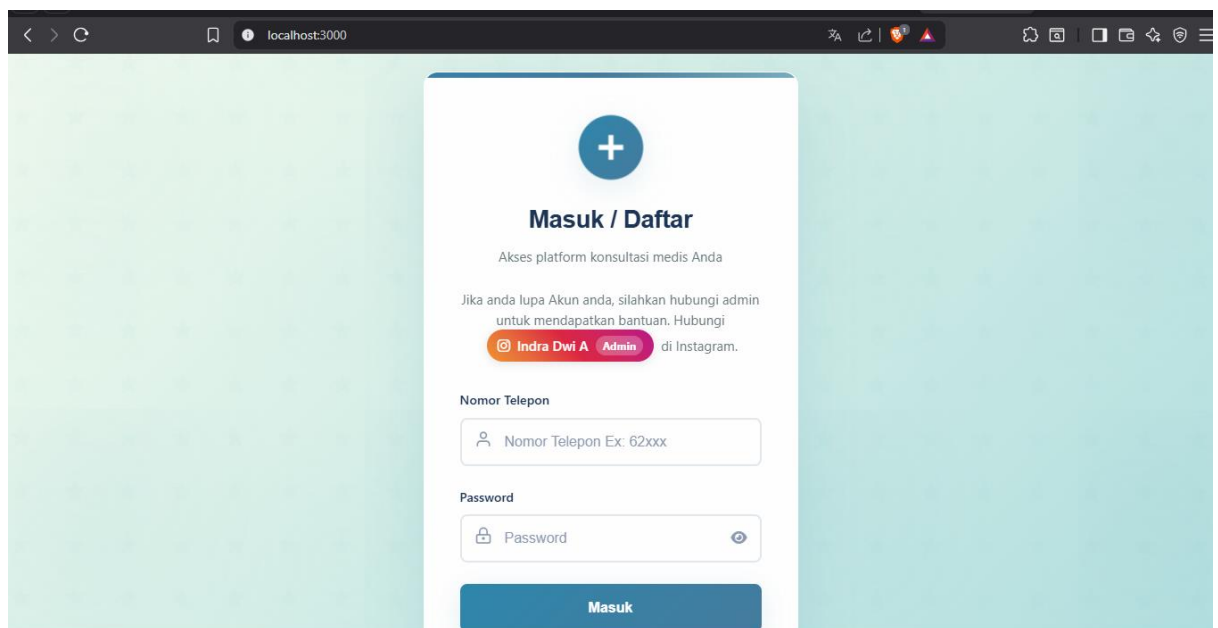
- Silakan di CMD tersebut ketikan “npm install .” atau “npm install” tanpa kutip
- Silakan tunggu sekitar 5 sampai 10 menit lebih, karena kita sedang melakukan penginstalan library atau perpustakaan, yang dimana ini menjadi syarat penting dalam menjalankann sebuah aplikasi dari nodejs, terutama yang menggunakan pustaka / perpustakaan / library. Seperti Axios, node-fetch, dan lain lain
- Setelah “npm install” maka kita jalankan file javascript indeks.js atau bisa gunakan launcher.js
- Silakan ketik “node launcher.js” atau “node .”
- Lalu ketik “y” untuk login (tanpa kutip)
- Lalu masukan nomor telepon yang diawali dengan 62. Contoh: 628xxxxx
- Setelah itu masukan kode pairing whatsapp di whatsapp hp android kita yang berada di “perangkat tertaut” lalu “tautkan perangkat” lalu pilih “tautkan dengan nomor telepon saja” setelah itu sudah selesai dan sudah jadi bot whatsappnya, dan juga server websitenya juga sudah menyala dengan ip yang berbeda namun port 3000. Contoh:
  - Server berjalan di: (Untuk IP itu tergantung dari internet nya)
  - - Localhost: <http://localhost:3000>
  - - LAN IP: <http://169.254.230.64:3000>
  - - LAN IP: <http://192.168.11.145:3000>
  - - LAN IP: <http://172.28.16.1:3000>

```
C:\Windows\System32\cmd.exe X + v
Microsoft Windows [Version 10.0.19045.6036]
(c) Microsoft Corporation. All rights reserved.

E:\MPI>node .
Server berjalan di:
- Localhost: http://localhost:3000
- LAN IP: http://169.254.230.64:3000
- LAN IP: http://192.168.11.145:3000
- LAN IP: http://172.28.16.1:3000
Jika ingin mengganti sesi, silahkan hapus folder auth dan jika ingin menghapus semua data, silahkan hapus db.sqlite
⚠ The printQRInTerminal option has been deprecated. You will no longer receive QR codes in the terminal automatically. Please listen to the connection.update event yourself and handle the QR your way. You can remove this message by removing this option. This message will be removed in a future version.
```

Gambar 3

Dan masukan diantara link tersebut ke browser, dan jadi deh, begini hasilnya



Gambar 4 Tampilan web

Selamat menggunakan, sekian dan terima kasih

NOTE: Instruksi dan cara penggunaan tidak ada yang salah, karena saya (Indra) mencobanya lebih awal dengan menggunakan virtualbox (dari kosong menjadi ada)