**PROPOSAL**

**MANAJEMEN PROYEK INFORMATIKA**

**PEMBUATAN APLIKASI WEBSITE MEDICAL CONVERSATION**

****

Oleh :

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

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

**2025**

* 1. Latar Belakang

**BAB I PENDAHULUAN**

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. Rumusan Masalah

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

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

Adapun tujuan dari penelitian ini adalah:

1. Membangun aplikasi website yang mendukung komunikasi antara pasien dan tenaga medis secara real-time.
2. Merancang fitur-fitur penting seperti live chat, unggah dokumen medis, dan manajemen riwayat konsultasi.
3. Menerapkan sistem keamanan data yang menjamin privasi dan kerahasiaan informasi pengguna.
   1. Manfaat Penelitian

Penelitian ini memberikan beberapa manfaat, antara lain:

1. Bagi Pasien: Mempermudah akses awal ke layanan medis secara daring.
2. Bagi Tenaga Medis: Membantu dalam melakukan penyaringan pasien berdasarkan gejala awal.
3. Bagi Peneliti: Menjadi referensi dalam pengembangan aplikasi berbasis kesehatan.
4. Bagi Masyarakat Umum: Meningkatkan kesadaran terhadap layanan kesehatan digital.
   1. Ruang Lingkup

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.

**BAB II TINJAUAN PUSTAKA**

* 1. Pengertian Sistem Informasi

Menurut Jogiyanto (2005), sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi serta menyediakan laporan-laporan yang dibutuhkan.

* 1. Aplikasi Berbasis Web

Aplikasi web adalah aplikasi yang diakses melalui peramban internet menggunakan jaringan. Aplikasi ini bersifat fleksibel dan dapat diakses dari berbagai perangkat tanpa instalasi tambahan. Dalam konteks layanan kesehatan, aplikasi web memberikan kemudahan akses bagi pasien dan tenaga medis di mana saja dan kapan saja.

* 1. Medical Conversation

Medical conversation adalah bentuk komunikasi digital antara pasien dan tenaga medis untuk membahas gejala, keluhan, dan arahan awal sebelum pemeriksaan lanjutan. Komunikasi ini bersifat non-diagnostik dan hanya bersifat penyaringan awal.

* 1. Studi Terkait

Beberapa aplikasi serupa telah dikembangkan seperti:

* + 1. Halodoc: Menyediakan layanan konsultasi dokter online berbasis aplikasi mobile.
    2. Alodokter: Menyediakan layanan tanya jawab dan artikel kesehatan berbasis web.

Penelitian ini berusaha membuat versi *light* dan *customizable* dari sistem serupa, yang dapat digunakan untuk klinik atau institusi kecil secara lebih fleksibel.

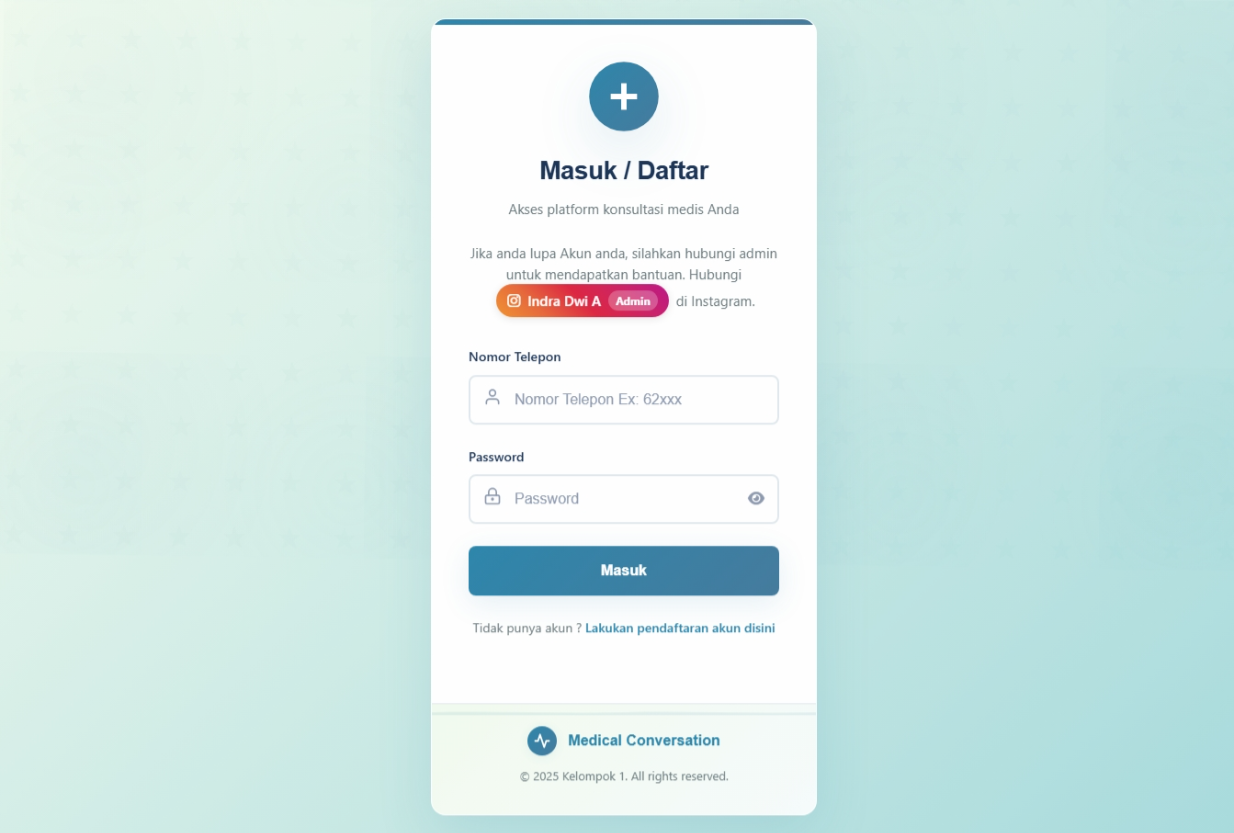
* 1. Kerangka Pemikiran

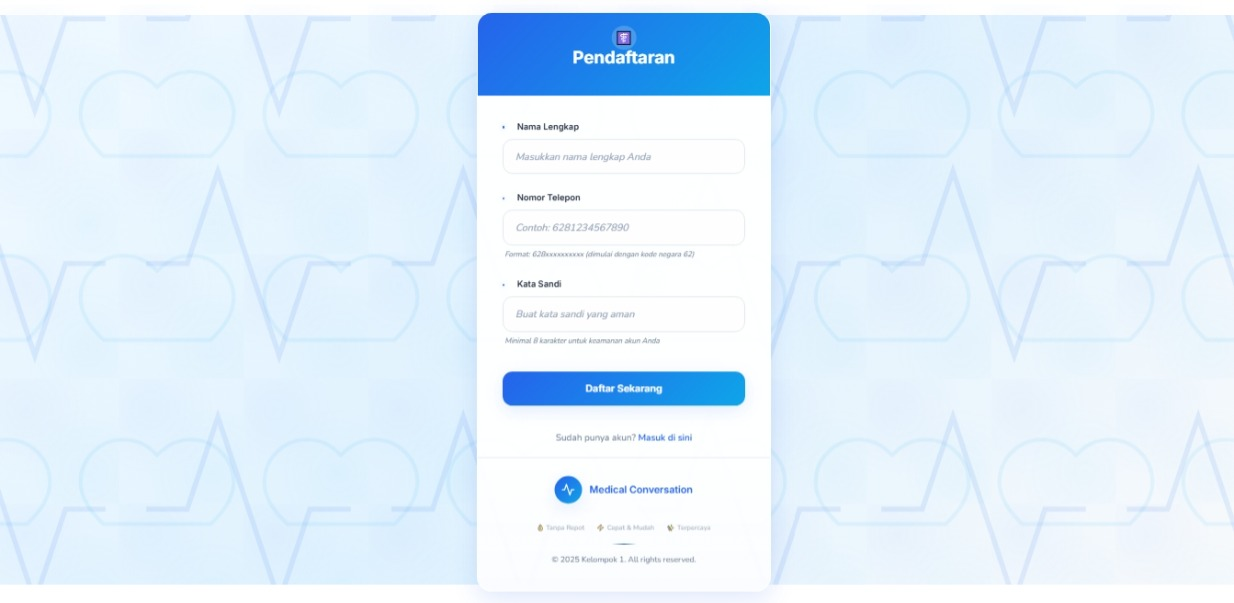
Gambaran umum pemikiran pengembangan sistem ini adalah:

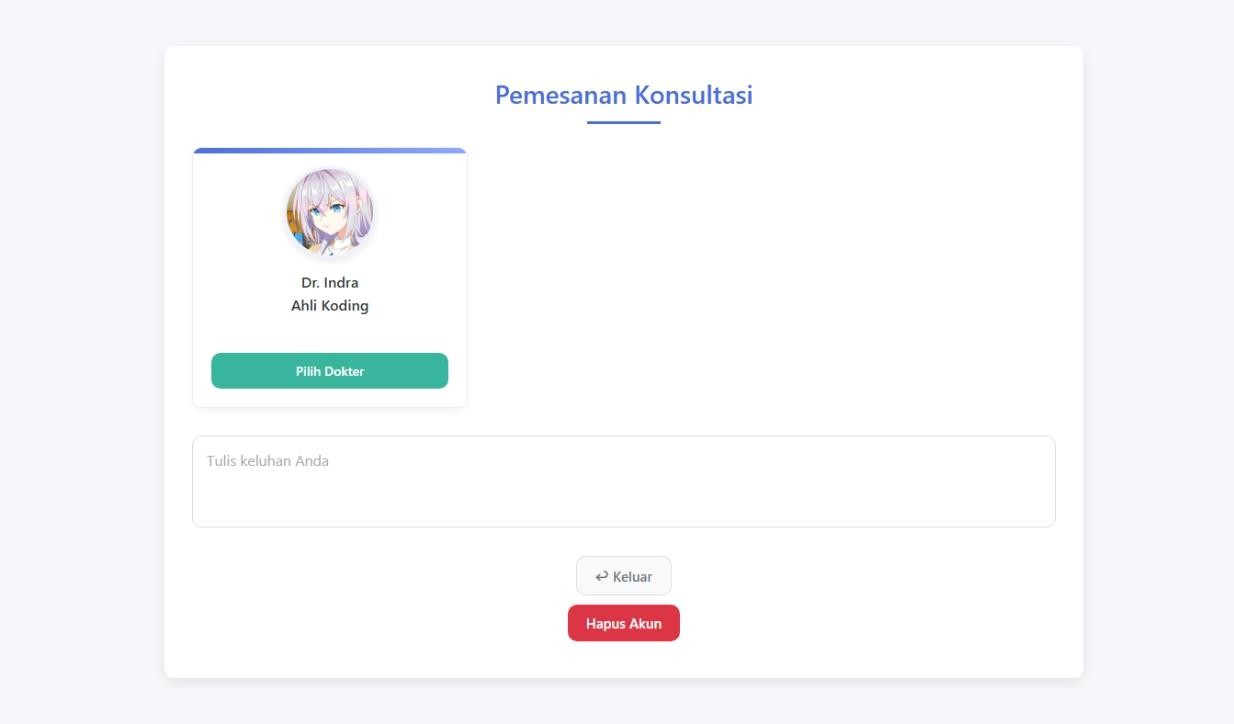
* + 1. Permasalahan komunikasi pasien-dokter → Solusi melalui aplikasi web.
    2. Diperlukan media percakapan → Fitur chat real-time.
    3. Privasi dan keamanan → Implementasi autentikasi dan enkripsi.

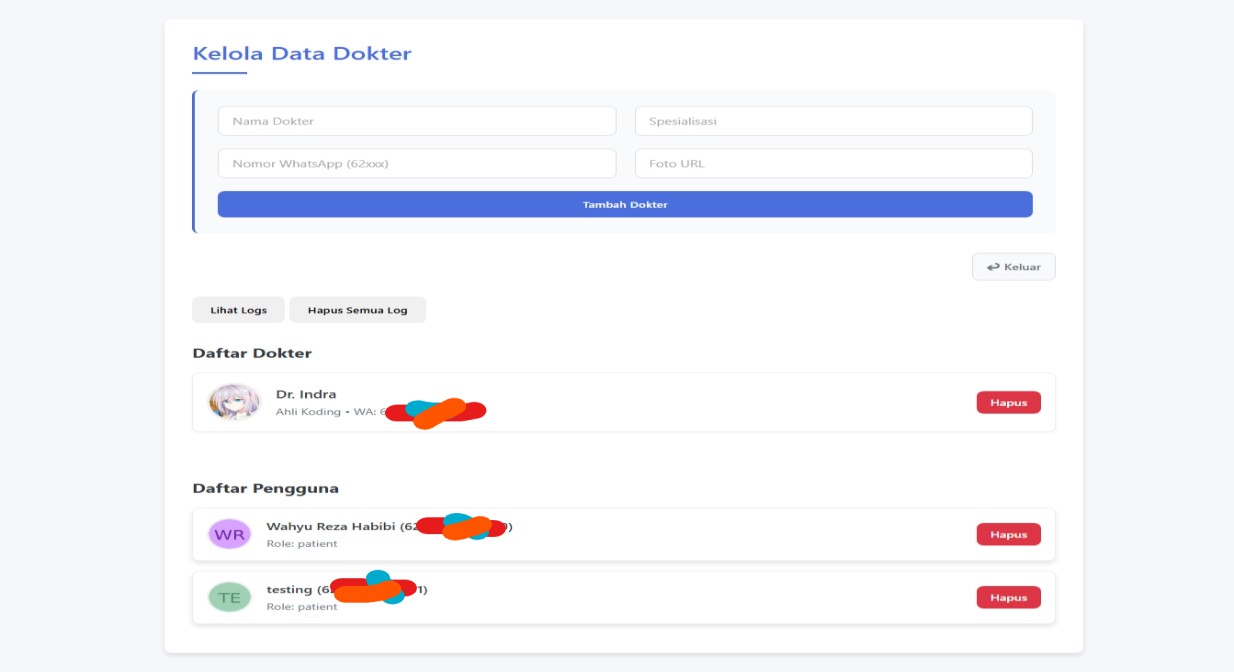
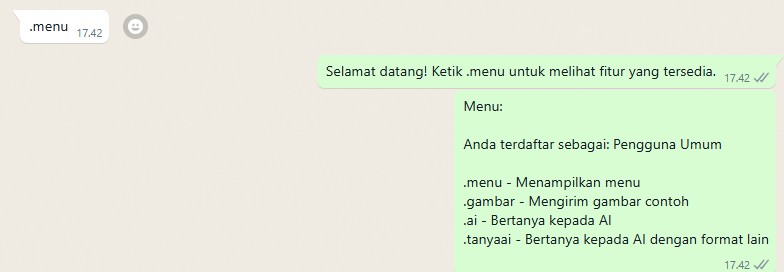
**BAB III**

**METODOLOGI PENELITIAN**

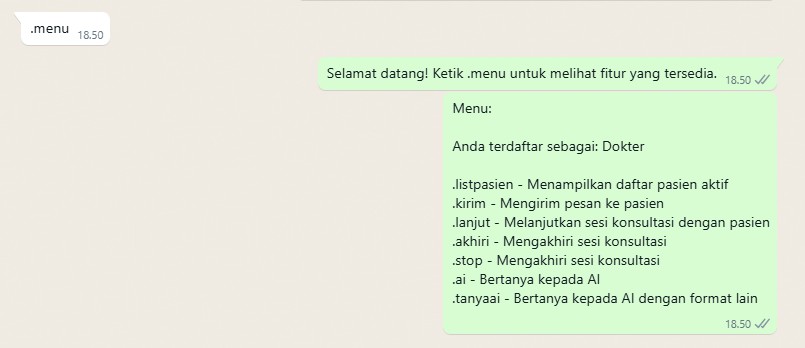
* 1. Ruang lingkup sistem
     1. Fronted
        1. UI/UX
           1. Halaman Login
           2. Halaman Pendaftaran



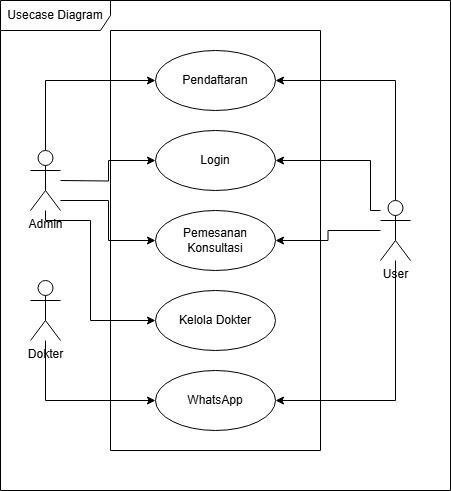
* + - * 1. Dashboard User
        2. Dashboard Admin



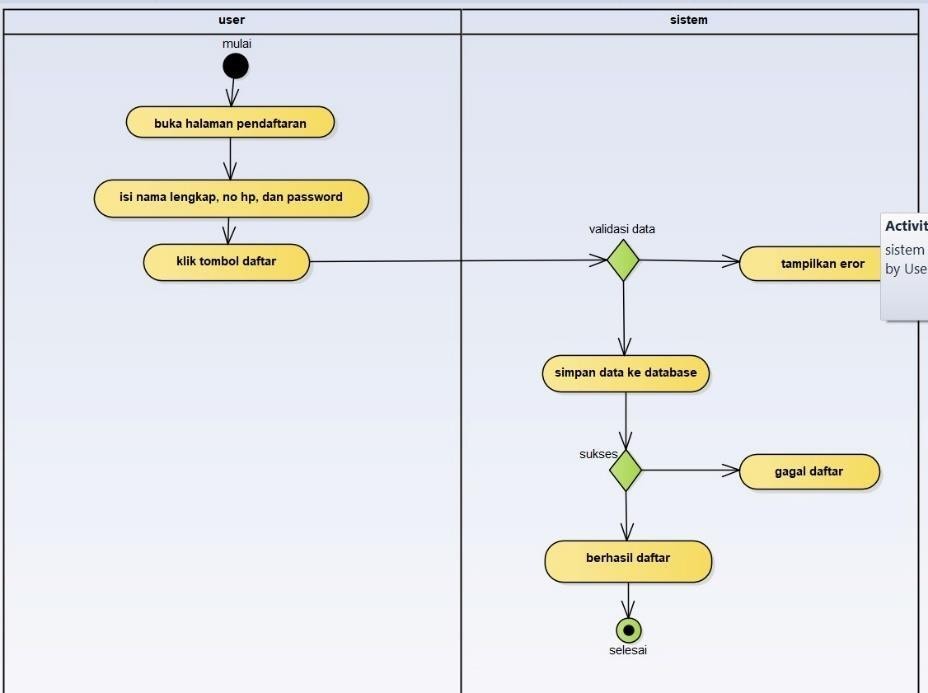
* + - * 1. Chatbot AI



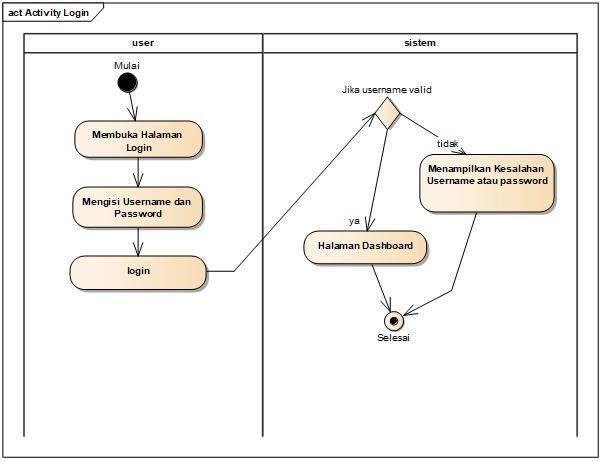
* + - 1. Use Case



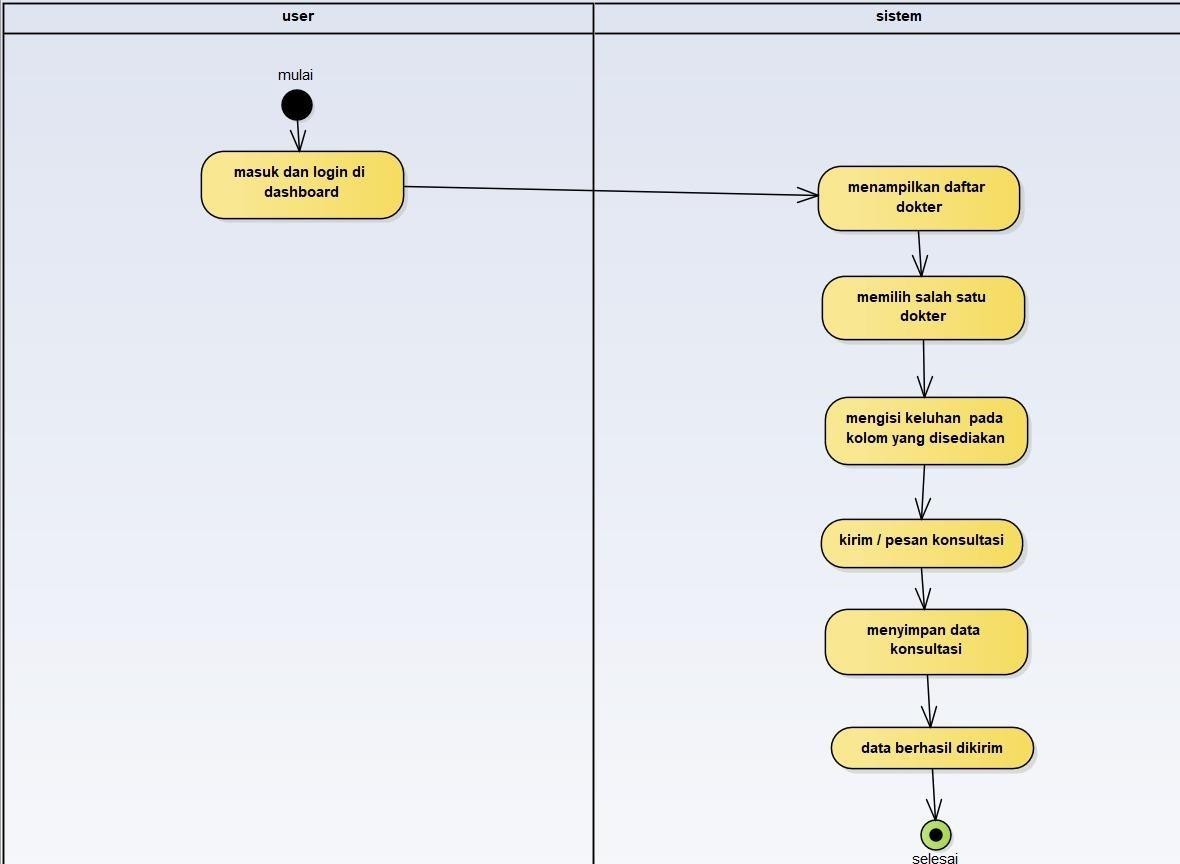
* + - 1. Activity Diagram
         1. Activity Pendaftaran

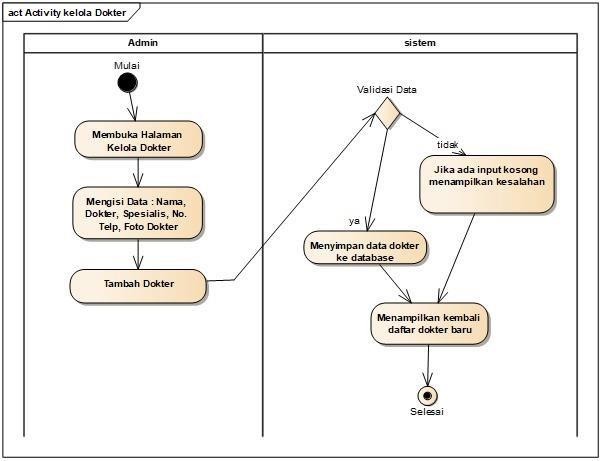


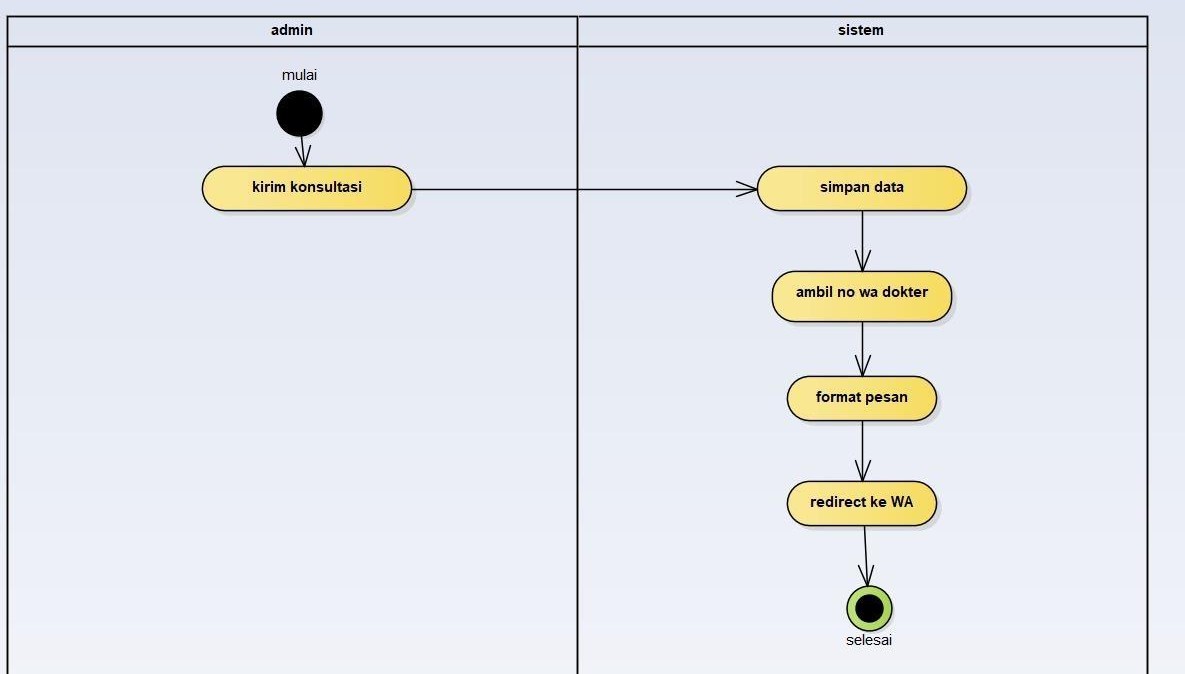
* + - * 1. Activity Login



* + - * 1. Activity Pemesanan Konsultasi

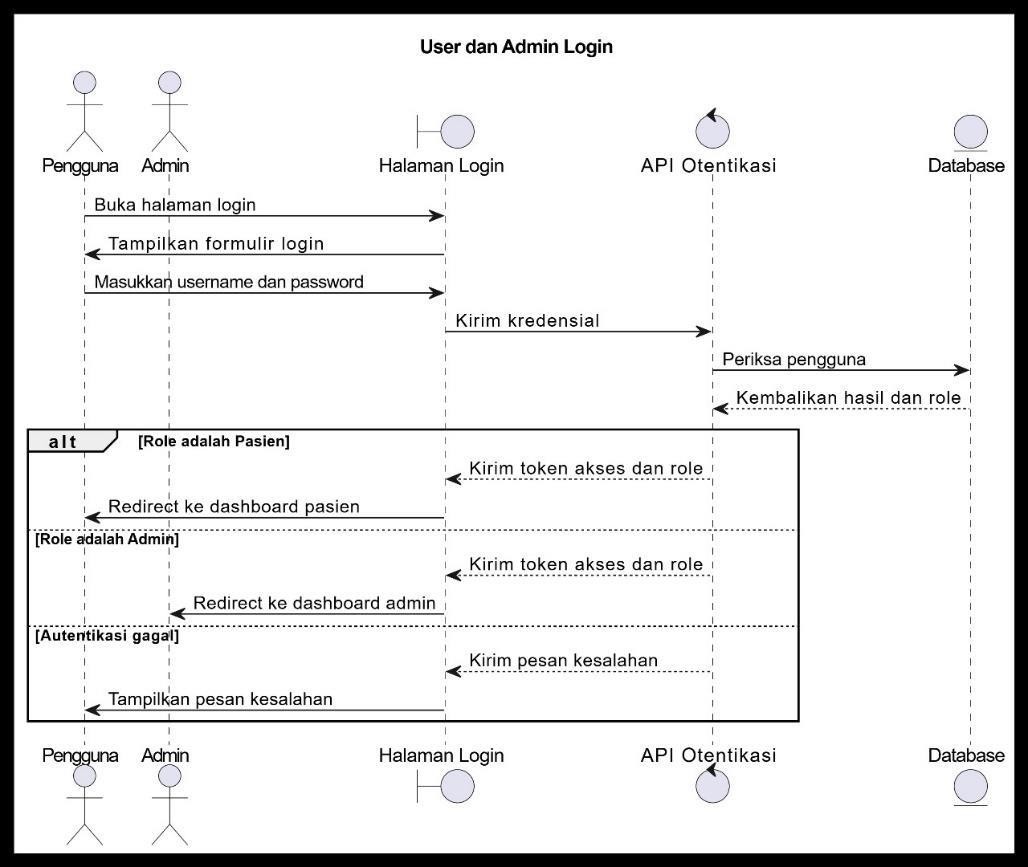


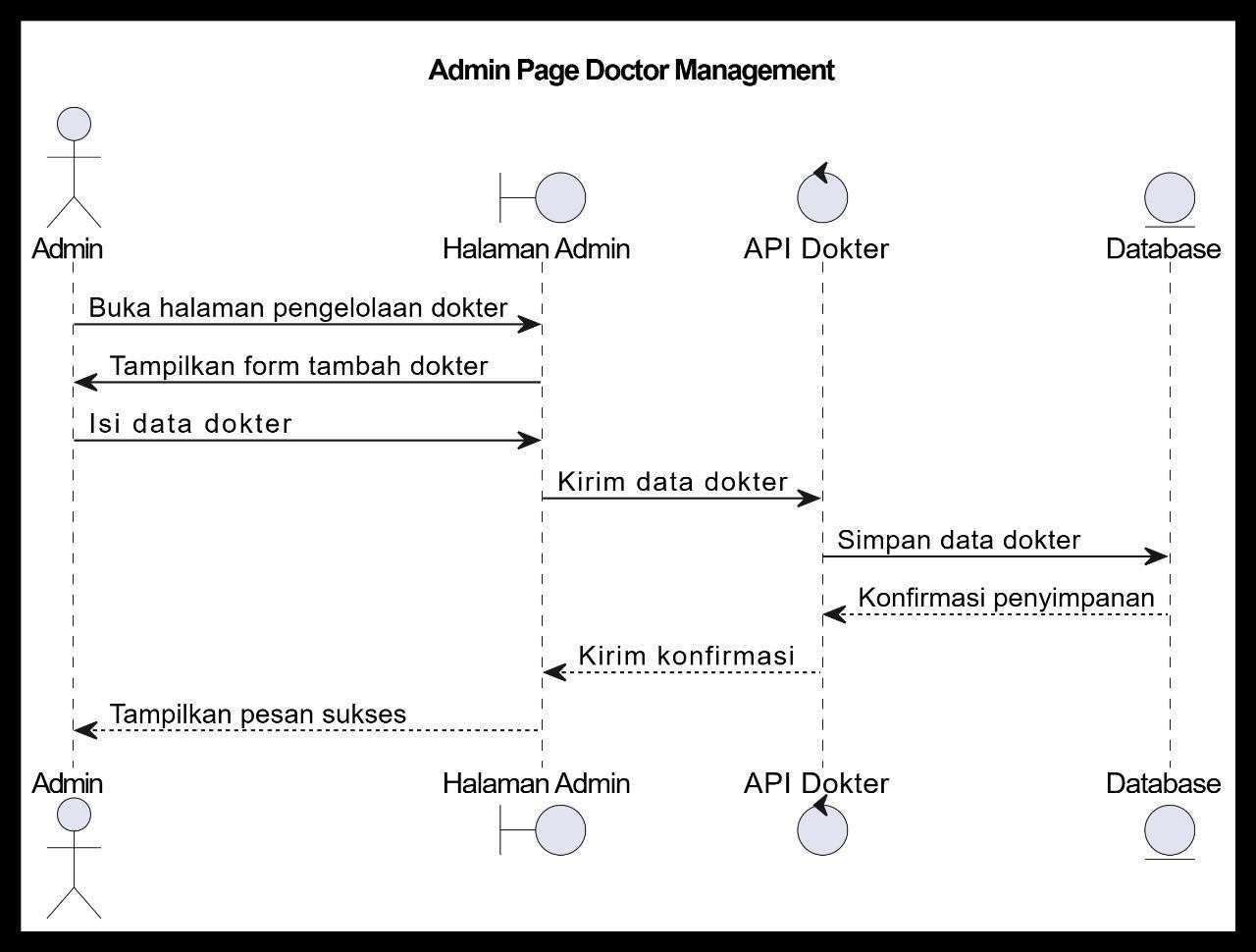
* + - * 1. Activity Kelola Dokter
        2. Activity WhatsApp



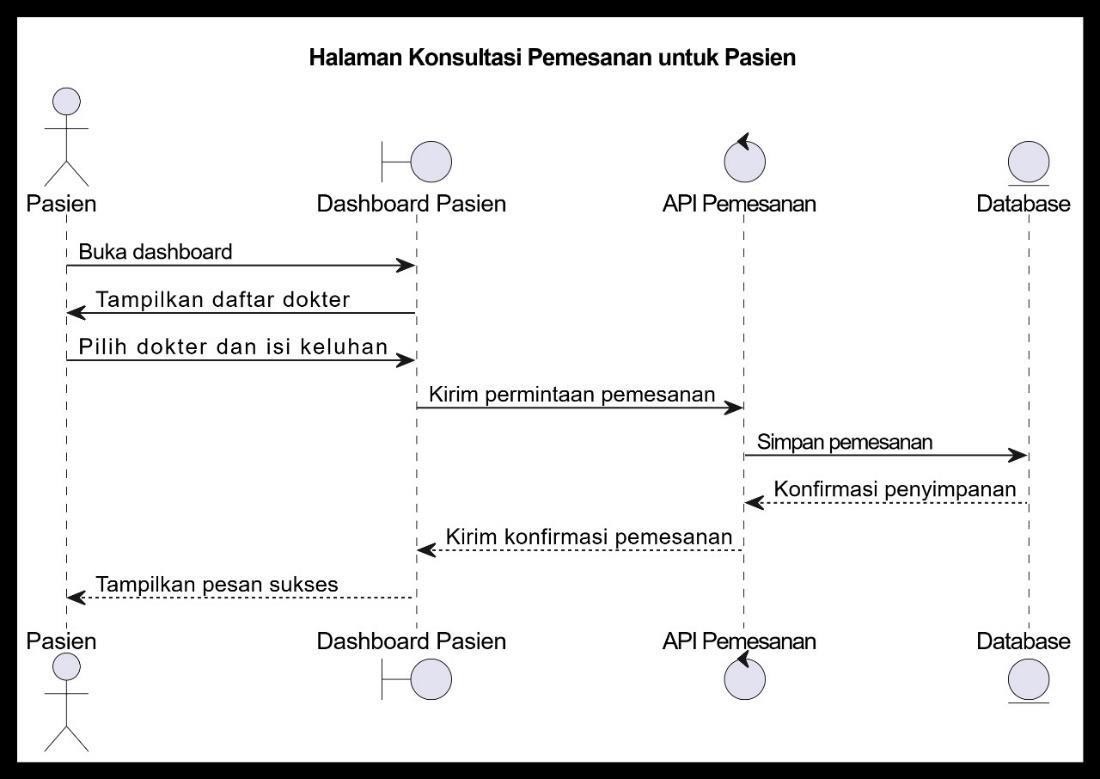
6. Activity Kode OTP



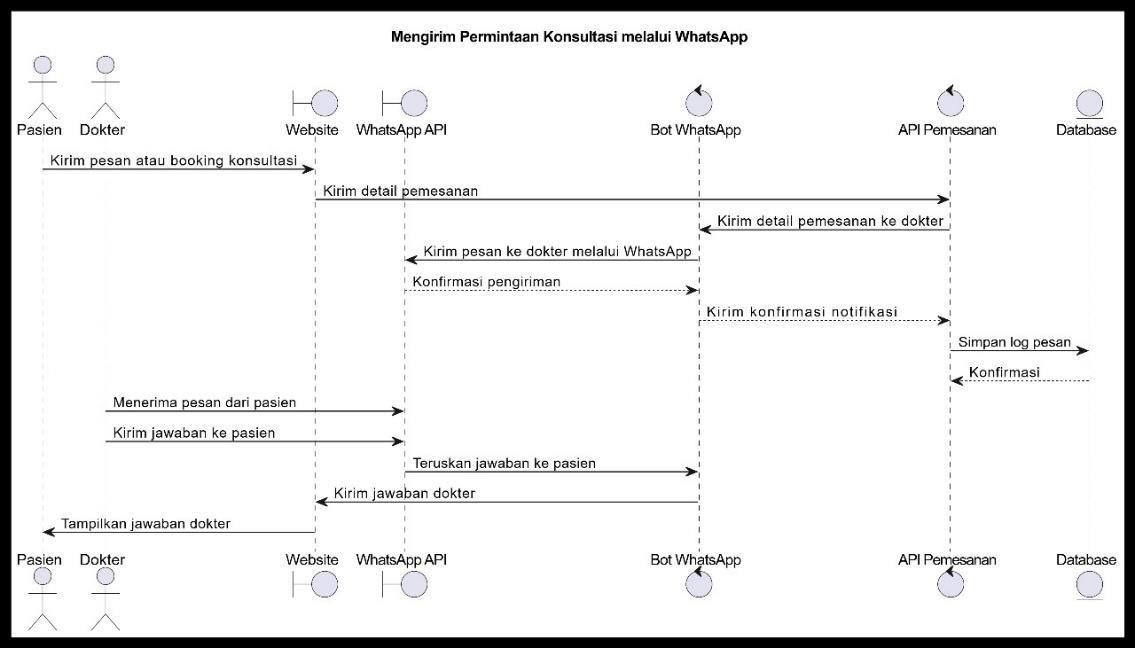
* + - 1. Sequence Diagram
         1. User dan Admin Login
         2. Kelola Dokter



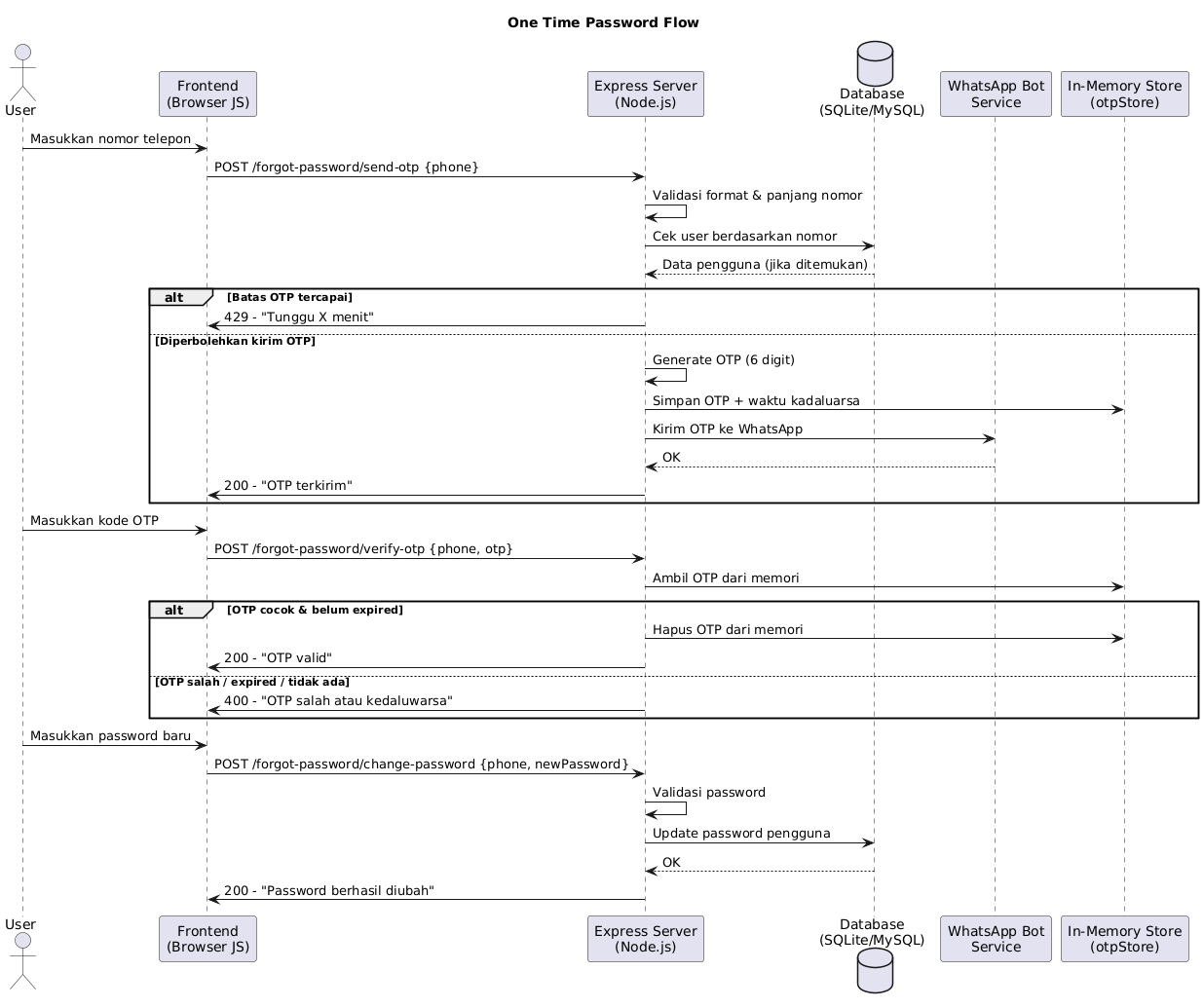
* + - * 1. Pemesanan Konsultasi

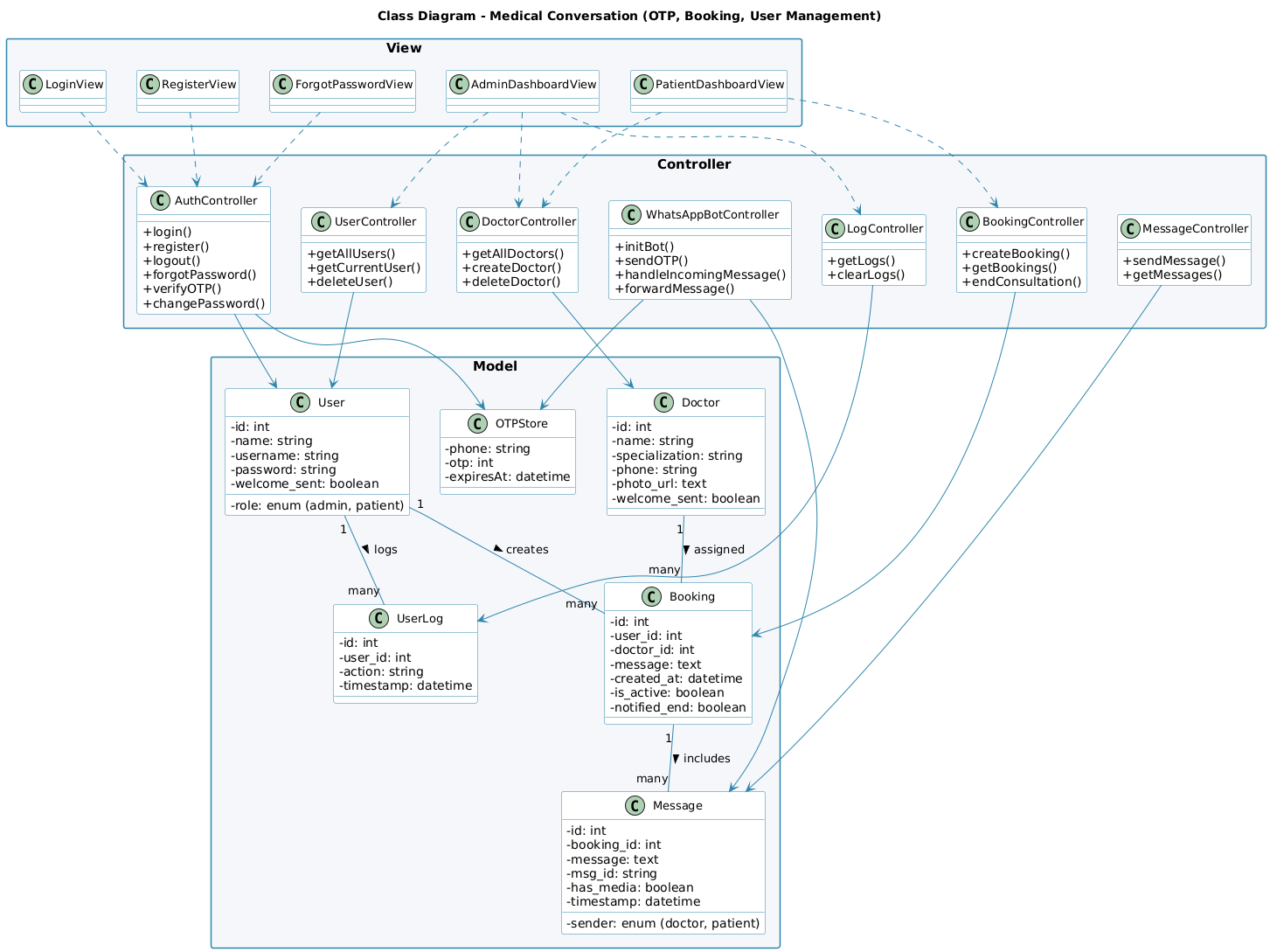


* + - * 1. Konsultasi Melalui WhatsApp



5. Kode OTP



* + - 1. Class Diagram
    1. Backend
       1. CDM (Conceptual Data Model)



* + - 1. LDM (Logical Data Model)



* + - 1. PDM CONVERSATION



* + - 1. SQL (Structued 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

);

* 1. Ruang Lingkup Proyek (WBS)

3.2.1 RAB ( Rencana Anggaran Biaya)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Kategori | Sub-kategori | Deskripsi | Biaya (Rp) |
| 1 | Pengembangan | Frontend | UI/UX, Login/Daftar, Dashboard, | 26.000.000 |
|  | Aplikasi | Development | Konsultasi, Manajemen Dokter |  |
| 2 |  | Pengembangan | Arsitektur, API (Otentikasi, | 17.000.000 |
|  |  | Backend | Pemesanan, Dokter, WhatsApp), |  |
|  |  |  | Integrasi DB |  |
| 3 |  | Pengembangan WhatsApp Bot | Pengembangan Logika Bot, Integrasi dengan WhatsApp API,  Pengembangan Fitur Pengiriman | 12.000.000 |
|  |  |  | Pesan |  |
| 4 | Desain | Desain UI/UX | Perancangan Antarmuka Pengguna, Prototyping | 5.000.000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | Infrastruktur Teknologi | Server dan Hosting | Sewa Server Backend, Hosting Website, Nama Domain | 5.000.000 |
| Basis Data | Manajemen dan Pengelolaan Database | 2.000.000 |
| 6 | Sumber Daya Manusia | Tim  Pengembang | Frontend  Developer/Backend/Fullstack/Mobile, DevOps | 15.000.000 |
| Tim Desain | UI/UX Designer | 3.500.000 |
| Tim  Manajemen Proyek | Manajer Proyek, Koordinator Tim | 8.000.000 |
| 7 | Biaya Lain-Lain | Biaya tak terduga |  | 5.000.000 |
| Biaya Administrasi |  | 2.000.000 |
| 8 | Pengujian dan Peluncuran | Pengujian | Pengujian Fitur dan Fungsionalitas, Pengujian Keamanan | 3.500.000 |
| Peluncuran | Persiapan dan Peluncuran Aplikasi | 1.500.000 |
| 9 | Pemeliharaan dan Dukungan | Pemeliharaan Rutin | Pemeliharaan dan Pembaruan | 3.000.000 |
|  |  | Dukungan Teknis | Dukungan Pelanggan dan Pengguna | 2.000.000 |
| 10 | Pelatihan dan Dokumentasi | Pelatihan Tim | Pelatihan Penggunaan Aplikasi | 1.000.000 |
| Dokumentasi | Pembuatan Dokumentasi Penggunaan | 1.000.000 |
| total | | 112.500.000 | | |

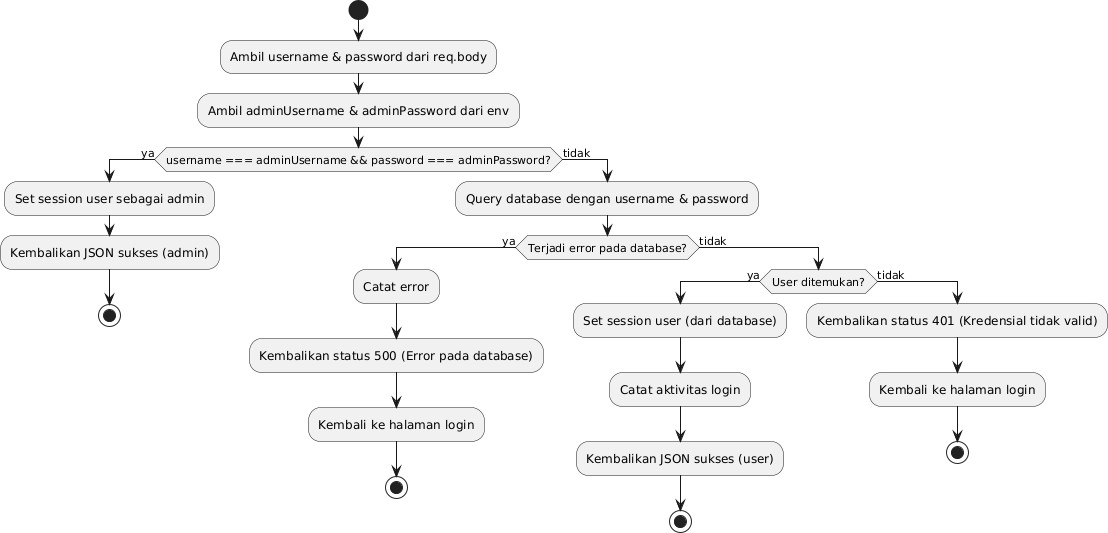
1. Pengujian
   1. Whitebox Testing
2. Login

**BAB IV PENGUJIAN ATAU TESTING**

|  |  |  |
| --- | --- | --- |
| **No.** | **Baris Kode** | **Deskripsi** |
| 1 | app.post('/login', (req, res) => { | Awal route handler POST  /login |
| 2 | const { username, password } = req.body; | Ambil username & password  dari request body |
| 3 | const adminUsername = process.env.ADMIN\_USERNAME; | Ambil admin username dari  environment |

|  |  |  |
| --- | --- | --- |
| 4 | const adminPassword = process.env.ADMIN\_PASSWORD; | Ambil admin password dari  environment |
| 5 | if (username === adminUsername && password === adminPassword) { | Cek apakah login sebagai  admin |
| 6 | req.session.user = { username: adminUsername, role: 'admin' }; | Set sesi user sebagai admin |
| 7 | return res.json({ status: 'success', role: 'admin' }); | Kirim respons sukses untuk  admin |
| 8 | } | Akhir blok if admin |
| 9 | db.get(SELECT \* FROM users WHERE username = ? AND  password = ?, [username, password], ... ) | Query database untuk login  user biasa |
| 10 | if (err) { | Cek jika ada error saat akses  database |
| 11 | console.error('Database error:', err); | Log error database |
| 12 | return res.status(500).json({ error: 'Database error' }); | Kirim respons error 500 jika  DB gagal |
| 13 | } | Akhir blok error |
| 14 | if (user) { | Cek apakah user ditemukan  dari query DB |
| 15 | req.session.user = { id: user.id, username: user.username, role: user.role }; | 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 |

|  |  |  |
| --- | --- | --- |
| **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 |

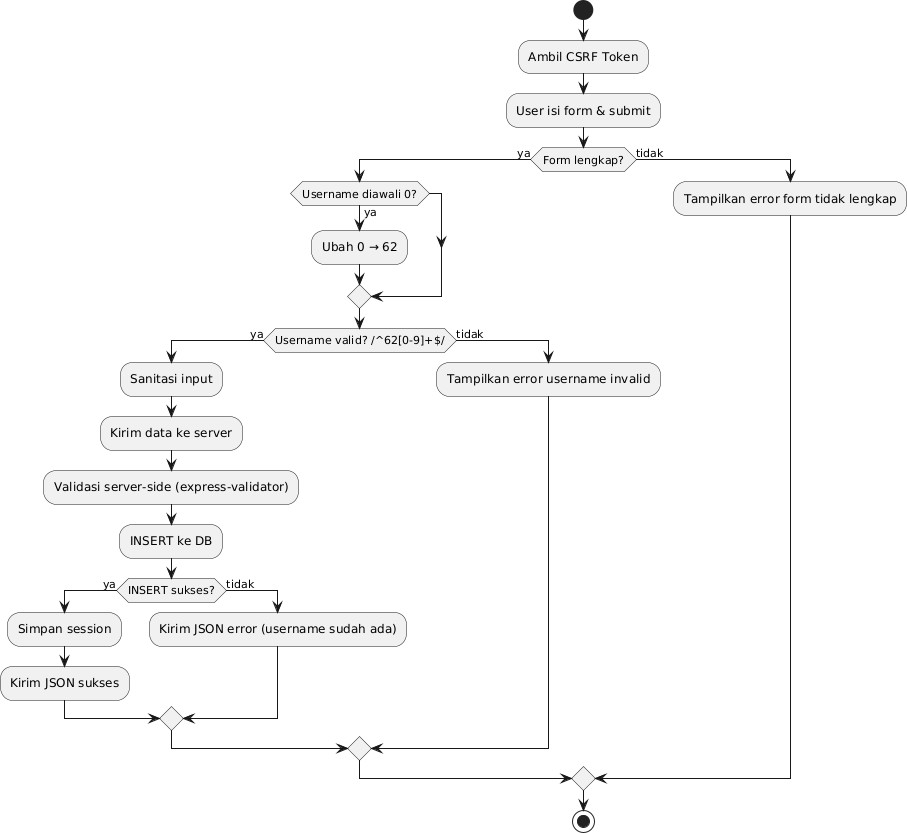


1. Register

|  |  |  |
| --- | --- | --- |
| **No**  **.** | **Baris Kode** | **Deskripsi** |
| 1 | app.post('/register', [...], (req, res)) | Awal route handler POST  /register |
| 2 | body('username').isAlphanumeric()... | Validasi username: alphanumeric |
| 3 | body('password').isLength({ min: 6 })... | Validasi password minimal 6  karakter |
| 4 | body('name').notEmpty()... | Validasi name tidak boleh kosong |
| 5 | (req, res) => { | Callback handler |
| 6 | const { username, password, name } = req.body; | Ambil data pendaftaran dari  request body |
| 7 | console.log(Mencoba...); | Log username yang akan  didaftarkan |
| 8 | db.run(INSERT INTO users..., [...], function (err) { | Query INSERT user ke database |
| 9 | if (err) { | Cek apakah terjadi error di DB |
| 10 | console.log('Gagal...'); | Log error |
| 11 | return res.status(400).json({ status: 'error', message: 'Username  already exists' }); | Kirim respons error |
| 12 | console.log('Pengguna berhasil...'); | Log sukses pendaftaran |
| 13 | req.session.user = { ... }; | Simpan data user di session |
| 14 | res.json({ status: 'success' }); | Kirim respons sukses |
| 15 | }); | Akhir callback DB |
| 16 | }); | Akhir route handler POST  /register |
| A1 | document.addEventListener('DOMContentLoaded'... | Ambil CSRF token dari server |
| A2 | document.getElementById('registerForm').addEventListener('su  bmit'...) | Tangani event submit |
| A3 | if (name === '' || username === '' || password === '') {  …..  }); | Validasi jika ada inputan kosong |

|  |  |  |
| --- | --- | --- |
| A4 | if (username.startsWith('0')) { ... } | Validasi: jika username diawali 0  → ganti 62 |
| A5 | const phoneRegex = /^62[0-9]+$/; if  (!phoneRegex.test(username)) { ... } | Validasi: username harus mulai  62 dan hanya angka |
| A6 | username, password, name = DOMPurify.sanitize(...) | Sanitasi data untuk mencegah  XSS / injeksi |
| A7 | fetch('/register', { method: 'POST', ... }) | Kirim data ke server via AJAX |
| A8 | if (data.status === 'success') { ... } | Tanggapi jika server merespons  sukses |
| A9 | else { ... } | Tanggapi jika server merespons  error |

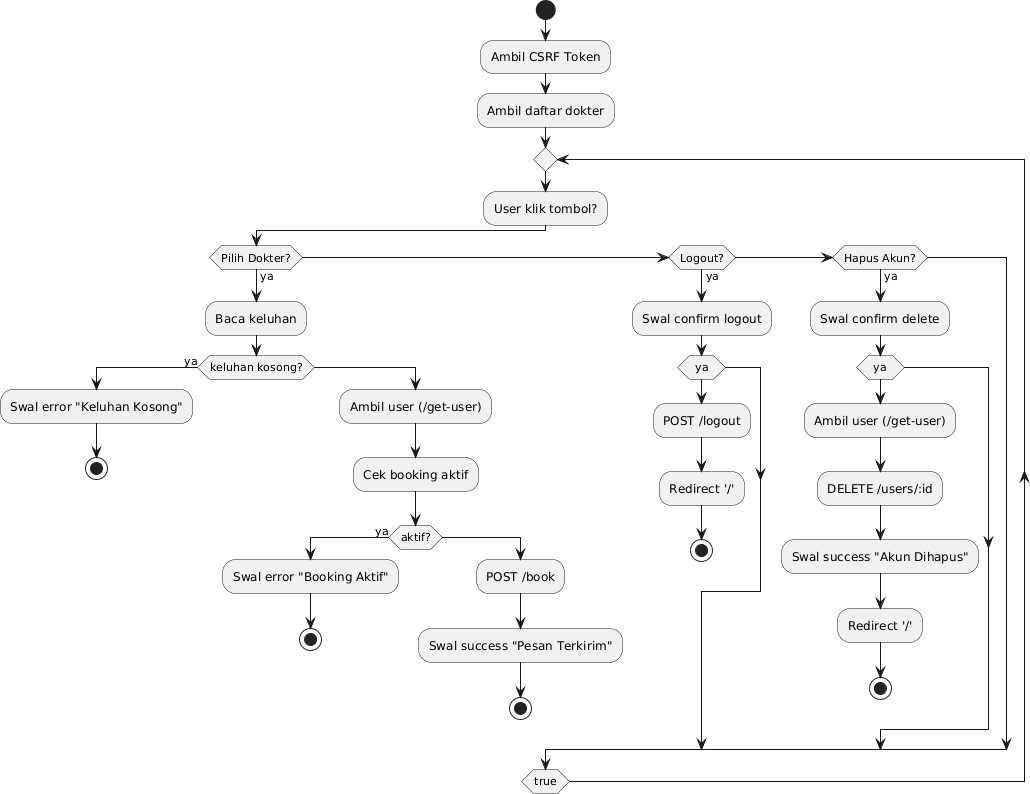
|  |  |  |
| --- | --- | --- |
| **Path**  **ID** | **Baris yang dilewati** | **Keterangan** |
| P1 | A1 → A2 → A3 (false) → A4 (false) → A5 (true) → A6 → A7 → 1  → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 (false) → 12 → 13 → 14 | Pengguna berhasil mendaftar (username valid, data disimpan  di DB) |
| P2 | A1 → 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 |

\

1. Dashboard User atau pasien

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

|  |  |  |
| --- | --- | --- |
| **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) |
| **P4** | 1 → 9 → 6 → 10 | Hapus akun sukses (konfirmasi → ambil user → delete →  redirect) |
| **P5** | 1 → 11 → 12 | Logout sukses (konfirmasi → POST logout → redirect) |



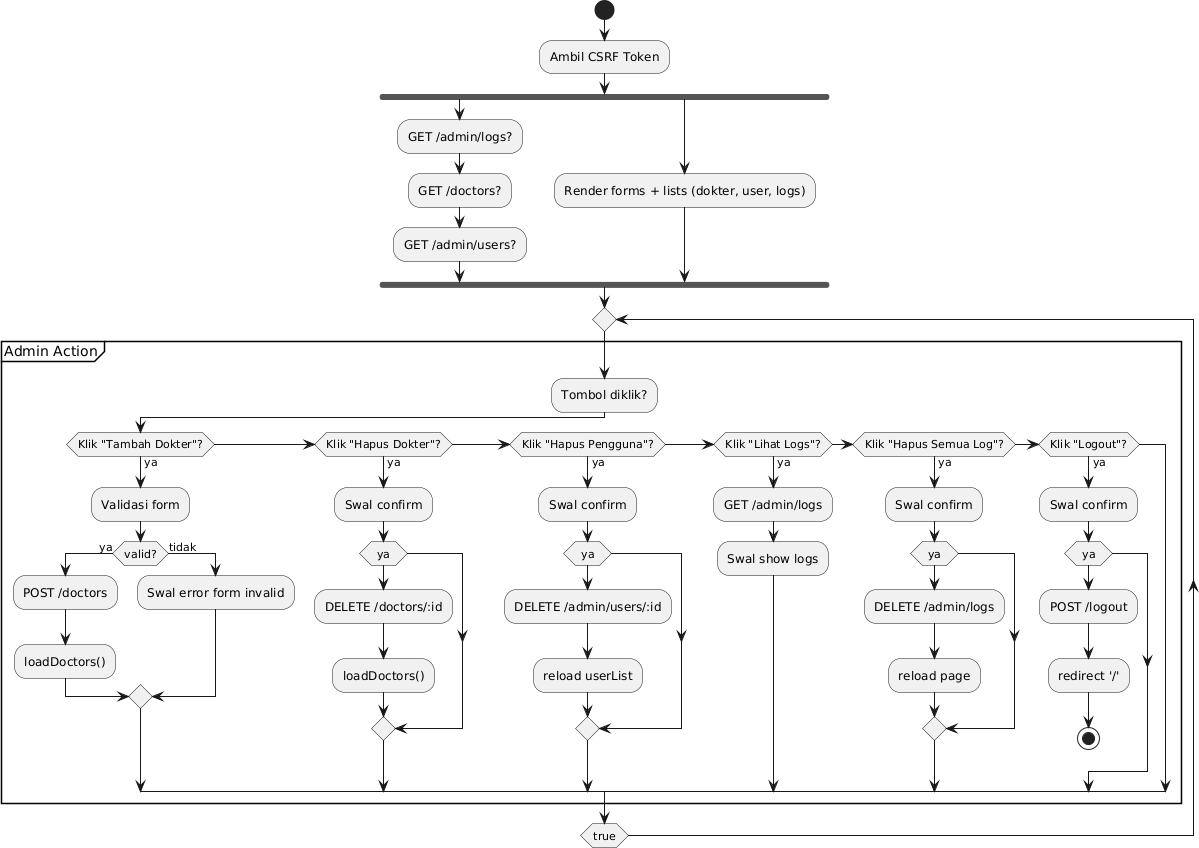
1. Dashboard Admin

|  |  |  |
| --- | --- | --- |
| **No**  **.** | **Baris Kode / Fungsi** | **Deskripsi** |
| 1 | app.get('/csrf-token', (req, res) => {<br> // Mengembalikan token CSRF<br> res.json({ csrfToken: req.csrfToken() });<br>}); | Route mengembalika  n CSRF token |
| 2 | app.use((err, req, res, next) => {<br> if (err.code === 'EBADCSRFTOKEN') {<br> // Token CSRF tidak valid<br> res.status(403).json({ error: 'Invalid CSRF token'  });<br> } else {<br> next(err);<br> }<br>}); | Tangani token  CSRF tidak valid (403) |

|  |  |  |
| --- | --- | --- |
| 3 | app.get('/', (req, res) => {<br> res.sendFile(path.join( dirname, 'public/index.html'));<br>}); | Kirim  index.html |
| 4 | app.use((err, req, res, next) => {<br> console.error(err);<br> res.status(500).json({ message: 'Internal Server Error' });<br>}); | Tangani semua error tak  terduga (500) |
| 5 | app.get('/admin/logs', isAuthenticated && isAdmin, (req, res) => {<br> db.all(`SELECT l.id, u.username, l.action, l.timestamp<br> FROM user\_logs l<br> JOIN users u ON l.user\_id = u.id<br> ORDER BY l.timestamp DESC`, [], (err, rows) => {<br> if (err) {<br> return res.status(500).json({ error: 'Gagal mengambil  log aktivitas' });<br> }<br> res.json(rows);<br> });<br>}); | Ambil semua log aktivitas (JOIN users + user\_logs) |
| 6 | app.delete('/admin/logs', isAuthenticated && isAdmin, (req, res) => {<br> db.run(`DELETE FROM user\_logs`, [], function (err) {<br> if (err) {<br> console.error('Error deleting logs:', err);<br> return res.status(500).json({ error: 'Gagal menghapus log aktivitas' });<br> }<br> res.json({ status: 'success', message:  'Semua log aktivitas berhasil dihapus' });<br> });<br>}); | Hapus seluruh log aktivitas |
| 7 | app.get('/doctors', isAuthenticated, (req, res) => {<br> if (!req.session.user) return res.status(403).json({ error: 'Unauthorized' });<br> db.all('SELECT \* FROM doctors', (err, rows) => {<br> if (err) return res.status(500).json({ error: 'Database error' });<br> res.json(rows);<br> });<br>}); | Ambil daftar dokter (cek session, DB  query) |
| 8 | app.post('/doctors', isAuthenticated && isAdmin, (req, res) => {<br> const { name, specialization, phone, photo\_url } = req.body;<br> if (!photo\_url.startsWith('http://') && !photo\_url.startsWith('https://')) {<br> return res.status(400).json({ error: 'Invalid photo URL' });<br> }<br> db.run(`INSERT INTO doctors (name, specialization, phone, photo\_url) VALUES (?, ?, ?, ?)`,<br> [name, specialization, phone, photo\_url], err => {<br> if (err) return res.status(500).json({ error: 'Gagal tambah  dokter' });<br> res.json({ status: 'success' });<br> });<br>}); | Tambah dokter baru (validasi URL foto, DB INSERT) |
| 9 | app.delete('/doctors/:id', isAuthenticated && isAdmin, (req, res) => {<br> const id = req.params.id;<br> db.run(`DELETE FROM doctors WHERE id = ?`, [id], function(err) {<br> if (err) return res.status(500).json({ error: 'Gagal hapus dokter'  });<br> res.json({ status: 'deleted' });<br> });<br>}); | Hapus dokter berdasarkan ID |
| 10 | app.delete('/admin/users/:id', isAuthenticated && isAdmin, (req, res) => {<br> const userId = req.params.id;<br> db.run(`DELETE FROM users WHERE id = ?`, [userId], function (err) {<br> if (err) return res.status(500).json({ error: 'Gagal menghapus akun pengguna' });<br> logUserActivity(`${userId}`,  'delete\_account\_by\_admin');<br> res.json({ status: 'success', message: `Akun dengan ID ${userId} berhasil dihapus` });<br> });<br>}); | Hapus user oleh admin (DB DELETE +  logUserActivity  ) |
| 11 | app.get('/admin/users', isAuthenticated && isAdmin, (req, res) => {<br> db.all(`SELECT id, username, name, role FROM users`, (err, rows) => {<br> if (err) return res.status(500).json({ error: 'Gagal mendapatkan daftar pengguna' });<br> res.json(rows);<br> });<br>}); | Ambil daftar semua user |
| 12 | app.post('/logout', (req, res) => {<br> logUserActivity(req.session.user.id, 'logout');<br> req.session.destroy(err => {<br> if (err) return res.status(500).json({ error: 'Gagal logout' });<br> res.json({ status: 'success', message: 'Logged out successfully' });<br> });<br>}); | Logout user (destroy session + logUserActivity  ) |
| 13 | fetch('/csrf-token')<br> .then(res => res.json())<br> .then(data => { document.getElementById('csrfToken').value = data.csrfToken; }); | Ambil dan simpan CSRF token di  #csrfToken |
| 14 | fetch('/admin/logs')<br> .then(res => res.json())<br> .then(data => Swal.fire({ html: logsHtml }));<br><br>fetch('/admin/logs', { method: 'DELETE', headers: { 'X-CSRF- Token': csrfToken } }) | Flow “Lihat Logs” & “Hapus Semua Log” (Swal confirm +  fetch) |

|  |  |  |
| --- | --- | --- |
| 15 | function loadDoctors() {<br> fetch('/doctors')<br> .then(res => res.json())<br>  .then(data => { /\* render list + tombol Hapus \*/ });<br>} | Render daftar  dokter, tombol hapus dokter |
| 16 | document.getElementById('addDoctorForm').addEventListener('submit', e => {<br> e.preventDefault();<br> fetch('/doctors', { method: 'POST', headers: { 'X-CSRF- Token': csrfToken }, body: JSON.stringify({ name, specialization, phone, photo\_url  }) })<br>}); | Tambah dokter via form (validasi  HTML5 + Swal) |
| 17 | function deleteDoctor(id) {<br> fetch(`/doctors/${id}`, { method: 'DELETE', headers:  { 'X-CSRF-Token': csrfToken } })<br> .then(() => loadDoctors());<br>} | Hapus dokter via button “Hapus” + CSRF  header |
| 18 | fetch('/admin/users')<br> .then(res => res.json())<br> .then(users => { /\* render userList + tombol Hapus \*/ }); | Tampilkan daftar user + tombol hapus  pengguna |
| 19 | function deleteUser(userId) {<br> fetch(`/admin/users/${userId}`, { method: 'DELETE', headers: { 'X-CSRF-Token': csrfToken } })<br> .then(() => location.reload());<br>} | Hapus user via Swal confirm +  CSRF header |

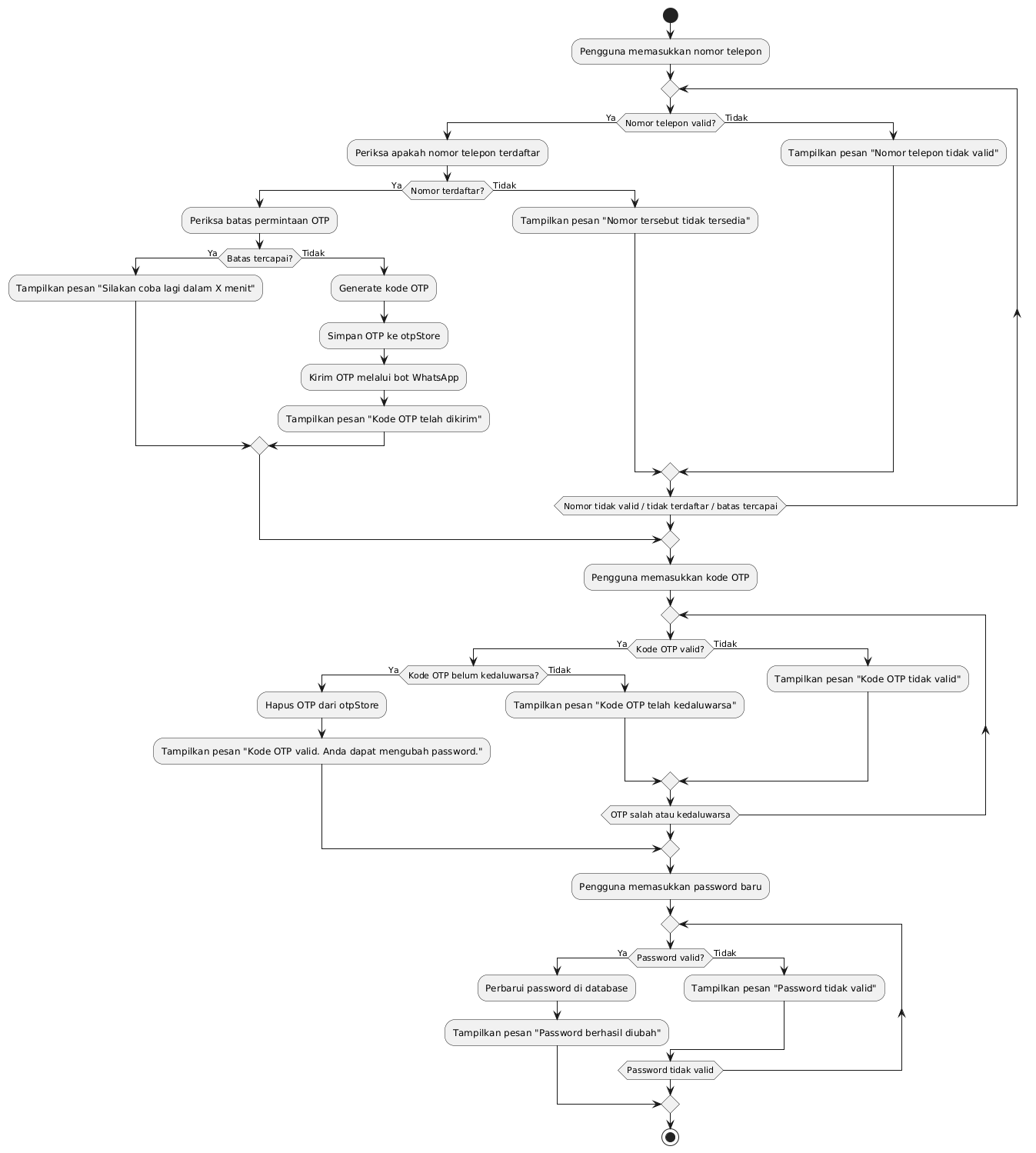
|  |  |  |
| --- | --- | --- |
| **Path**  **ID** | **Baris yang Dilewati** | **Keterangan** |
| **P1** | 1 → 3 → 7 → 15 →  16 | Tambah dokter sukses (GET csrf, loadDoctors, submit form valid) |
| **P2** | 1 → 3 → 7 → 15 →  17 | Hapus dokter sukses (GET csrf, loadDoctors, Swal confirm, DELETE) |
| **P3** | 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 |



E. OTP (One Time Password)

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

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



* 1. Blackbox Testing

1. 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 |

1. 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 |

1. 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 Terpillih | 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 |

1. 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)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 valid (misalnya terlalu pendek) | Muncul pesan "Password tidak valid." | Muncul pesan "Password tidak valid." | Berhasil |

**BAB V 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**)

2. Lingga Safitri (**Project Manager, Tester**)

3. Prayoga Pratama (**Project Manager, Advisor/Penasihat/Perekomendasian**)

4. Muhammad Rizki Ramadhan (**Project Manager, Tester**)

|  |  |  |  |
| --- | --- | --- | --- |
| 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, kalaupun 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. Sebenernya 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 {  error:"unauthorized" }  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 nnti 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, Spesialisasi-nya, Nomor Whatsapp, dan Foto Dokter (menggunakan url website atau disimpan di server lalu dipanggil direktori folder gambarnya yaitu di http://localhost:3000/images/ yang dimana akan menampilkan json link url tersebut dan akan menjadi seperti ini <http://localhost:3000/gambar/alya.jpg> 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. |