

**RANCANG BANGUN SISTEM INFORMASI MANAJEMEN  
KLINIK GIGI**

(Studi Kasus : Klinik Gigi Xenon Dental House)

**TUGAS AKHIR**

Diajukan Sebagai Salah Satu Syarat untuk Menyelesaikan Program Strata-1 pada  
Departemen Sistem Informasi Fakultas Teknologi Informasi Universitas Andalas

**Oleh :**

**Alif Abdul Rauf**

**2011522024**

**Pembimbing:**

**Husnil Kamil, MT**

**198201182008121002**



**DEPARTEMEN SISTEM INFORMASI  
FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS ANDALAS**

**2024**

## PERNYATAAN

Saya menyatakan bahwa laporan tugas akhir berjudul “**Rancang Bangun Sistem Informasi Manajemen Klinik Gigi (Studi Kasus: Klinik Gigi Xenon Dental House)**” ini merupakan karya asli saya dan sepanjang pengetahuan saya tidak ada karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain untuk memenuhi syarat tugas akhir di perguruan tinggi lain, kecuali yang secara tertulis diacu dalam naskah ini dan dicantumkan dalam daftar pustaka.

Padang, 30 Agustus 2024

Penulis

A handwritten signature in black ink, appearing to read 'Alif', with a stylized flourish at the end.

Alif Abdul Rauf

## **KATA PENGANTAR**

Puji syukur kehadiran Allah SWT atas rahmat dan karunia-Nya sehingga saya dapat menyelesaikan Laporan Tugas Akhir yang berjudul “Rancang Bangun Sistem Informasi Manajemen Klinik Gigi (Studi Kasus: Klinik Gigi Xenon Dental House)” sebagai salah satu syarat akademik untuk menyelesaikan mata kuliah tugas akhir di Departemen Sistem Informasi, Fakultas Teknologi Informasi, Universitas Andalas.

Dalam proses penyusunan tugas akhir ini, saya menerima banyak bantuan dan dukungan dari berbagai pihak. Oleh karena itu, saya ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Kedua orang tua dan keluarga yang selalu membantu, memberikan nasihat, semangat, dan doa dalam proses penyusunan tugas akhir ini.
2. Bapak Ricky Akbar, M.Kom., selaku Ketua Program Studi Sistem Informasi Universitas Andalas sekaligus pembimbing tugas akhir yang telah memberikan bimbingan selama proses penyusunan tugas akhir ini.
3. Bapak Husnil Kamil, M.T., selaku pembimbing tugas akhir sekaligus Wakil Dekan II Fakultas Teknologi Informasi Universitas Andalas yang telah memberikan bimbingan selama proses penyusunan tugas akhir ini.
4. Bapak Fajril Akbar, M.Sc., selaku dosen pembimbing akademik yang senantiasa memberikan semangat dan arahan selama masa studi di Sistem Informasi Universitas Andalas.
5. Teman-teman yang telah membantu dan mendampingi saya selama masa studi di Universitas Andalas yang tidak dapat disebutkan satu per satu.
6. Klinik Gigi Xenon Dental House serta seluruh pihak yang telah membantu dan memberikan dukungan dalam penyusunan tugas akhir ini.

Penulis menyadari bahwa laporan tugas akhir ini masih jauh dari sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang membangun dari para pembaca. Kritik dan saran tersebut dapat disampaikan melalui email: [alifabdulrauf@gmail.com](mailto:alifabdulrauf@gmail.com). Semoga laporan tugas akhir ini dapat bermanfaat bagi penulis dan para pembaca.

Padang, 30 Agustus 2024

Penulis,

A handwritten signature in black ink, appearing to be 'Alif' followed by a stylized surname.

Alif Abdul Rauf

## DAFTAR ISI

PERNYATAAN.....	ii
KATA PENGANTAR .....	iii
DAFTAR ISI.....	v
DAFTAR GAMBAR .....	vii
DAFTAR TABEL.....	x
ABSTRAK .....	1
BAB I PENDAHULUAN .....	1
1.1 Latar Belakang .....	2
1.2 Rumusan Masalah .....	5
1.3 Batasan Masalah.....	5
1.4 Tujuan Penelitian.....	5
1.5 Sistematika Penulisan.....	5
BAB II TINJAUAN PUSTAKA.....	7
2.1 Klinik Gigi.....	7
2.2 Klinik Gigi Xenon Dental House .....	9
2.3 Sistem Informasi Manajemen.....	10
2.4 <i>Progressive Web App (PWA)</i> .....	11
2.5 Alat Analisis dan Perancangan.....	12
2.5.1 Business Process Modelling Notation (BPMN) .....	12
2.5.2 Use Case Diagram .....	15
2.5.3 Entity Relationship Diagram(ERD).....	17
BAB III METODE PENELITIAN.....	19
3.1 Objek Penelitian .....	19
3.2 Metode Pengumpulan Data .....	20
3.2.1 Observasi .....	20
3.2.2 Wawancara.....	20
3.2.3 Analisis Dokumen.....	20
3.2.4 Studi Literatur .....	20
3.3 Metode Pengembangan .....	21
3.4 Flowchart Penelitian.....	23
BAB IV ANALISIS DAN PERANCANGAN .....	25
4.1 Analisis Sistem .....	25

4.1.1	Analisis sistem yang Sedang Berjalan .....	25
4.1.2	Sistem yang Diusulkan .....	28
4.1.3	Analisis Kebutuhan Fungsional .....	31
4.1.4	Use Case Diagram .....	33
4.1.5	Sequence Diagram .....	35
4.2	Perancangan sistem .....	38
4.2.1	Perancangan Basis data.....	38
4.2.2	Arsitektur Aplikasi.....	44
4.2.3	Class Diagram.....	46
4.2.4	Tampilan Mockup Antarmuka.....	50
BAB V IMPLEMENTASI DAN PENGUJIAN .....		55
5.1	Implementasi Sistem .....	55
5.1.1	Pengkodean sistem.....	55
5.1.2	Implementasi Antarmuka sistem .....	66
5.2	Pengujian Sistem .....	69
5.2.1	Fokus Pengujian.....	69
5.2.2	Kasus Hasil Pengujian .....	71
5.2.3	Kesimpulan hasil pengujian.....	89
5.2.4	Analisa dan Pembahasan Hasil .....	92
BAB VI KESIMPULAN DAN SARAN .....		96
6.1	Kesimpulan.....	96
6.2	Saran .....	97
DAFTAR PUSTAKA .....		98
LAMPIRAN A .....		102
LAMPIRAN B .....		114
LAMPIRAN C .....		118
LAMPIRAN D .....		134
LAMPIRAN E .....		154
LAMPIRAN F.....		206
LAMPIRAN G.....		213

## DAFTAR GAMBAR

Gambar 2.1 Struktur Organisasi Klinik Gigi Xenon Dental House .....	9
Gambar 2.2 Penerapan <i>PWA</i> pada <i>website</i> trivago.....	11
Gambar 2. 3 Notasi Swimlanes.....	13
Gambar 2. 4 Notasi Connecting Object.....	13
Gambar 2. 5 Notasi Start, Intermediate, dan End Event.....	14
Gambar 2. 6 <i>Manual, user, service, receive task</i> .....	14
Gambar 2. 7 Notasi Gateway .....	14
Gambar 2. 8 Notasi Artifacts .....	15
Gambar 2.9 ERD hubungan instruktur dan pelajar.....	17
Gambar 3.1 Waterfall Model .....	22
Gambar 3.2 Tahapan Penelitian.....	24
Gambar 4.1 BPMN Sistem reservasi .....	26
Gambar 4.2 BPMN Sistem pelayanan .....	27
Gambar 4.3 BPMN manajemen jadwal .....	28
Gambar 4.4 BPMN Sistem reservasi .....	29
Gambar 4.5 BPMN Sistem pelayanan yang diusulkan.....	30
Gambar 4.6 BPMN Perizinan dokter.....	31
Gambar 4.7 Use Case Diagram Sistem.....	34
Gambar 4.8 Sequence Diagram input data user.....	35
Gambar 4.9 Sequence diagram login.....	36
Gambar 4.10 Sequence diagram reservasi admin .....	37
Gambar 4.11 Sequence diagram lihat dashboard owner.....	37
Gambar 4.12 ERD Sistem Informasi Manajemen Klinik.....	38
Gambar 4.13 Arsitektur Aplikasi <i>Website</i> .....	44
Gambar 4.14 Class Diagram Konfirmasi Reservasi Dari Pasien.....	47
Gambar 4.15 Class Diagram Pengajuan Permintaan Izin Dokter.....	48
Gambar 4.16 Class Diagram Pemilik Memutuskan Perizinan Dokter .....	49
Gambar 4.17 Mockup Form Reservasi .....	50
Gambar 4.18 Mockup Tampilan data user .....	51
Gambar 4.19 Mockup Dashboard akun owner .....	52
Gambar 4.20 Mockup Landing page <i>website</i> .....	53

Gambar 4.20 <i>Mockup</i> Antarmuka <i>Offline PWA</i> .....	54
Gambar 5.1 Potongan Kode Route Dokter .....	56
Gambar 5.2 Potongan Kode Route admin .....	57
Gambar 5.3 Potongan kode DataDokterController.....	58
Gambar 5.4 Potongan kode DataDokterController.....	59
Gambar 5.5 Potongan kode model detail jadwal .....	61
Gambar 5.6 Potongan kode model dokter .....	62
Gambar 5.7 Potongan kode view datadokter_add .....	63
Gambar 5.8 Potongan kode View datadokter .....	64
Gambar 5.9 Kode serviceworker.js.....	65
Gambar 5.10 Kode manifest.json .....	65
Gambar 5.11 Implementasi Antarmuka Halaman Dashboard Owner .....	66
Gambar 5.12 Implementasi Antarmuka Dashboard Admin .....	67
Gambar 5.13 Implementasi Antarmuka Reservasi dari admin .....	68
Gambar 5.14 Implementasi Antarmuka Reservasi Pasien.....	68
Gambar 5.15 Tampilan awal reservasi dari admin .....	72
Gambar 5.16 Tampilan Form tambah reservasi pasien baru .....	72
Gambar 5.17 Tampilan reservasi setelah data berhasil ditambahkan.....	73
Gambar 5.18 Tampilan notifikasi saat input form reservasi tidak diisi.....	74
Gambar 5.19 Tampilan form registrasi akun .....	75
Gambar 5.20 Tampilan Setelah pasien berhasil register.....	75
Gambar 5.21 Penambahan data user setelah registrasi akun pasien dilakukan .....	76
Gambar 5.22 Notifikasi saat mendaftarkan akun email yang sudah ada .....	77
Gambar 5.23 Tampilan form saat pasien melakukan reservasi .....	78
Gambar 5.24 Tampilan history pasien selesai melakukan reservasi .....	78
Gambar 5.25 Tampilan user admin setelah pasien selesai melakukan reservasi .....	79
Gambar 5.26 Tampilan data reservasi admin setelah reservasi pasien diterima .....	79
Gambar 5.27 Tampilan <i>history</i> reservasi pasien setelah reservasi pasien diterima .....	79
Gambar 5.28 Tampilan history reservasi pasien setelah melakukan reservasi .....	80



Gambar 5.29 Tampilan reservasi pasien setelah melakukan reservasi pada admin .....	81
Gambar 5.30 Tampilan histori reservasi pasien setelah ditolak admin .....	81
Gambar 5.31 Tampilan landing page saat server aktif .....	82
Gambar 5.32 Tampilan saat server dimatikan .....	83
Gambar 5.33 Tampilan landing page tetap muncul setelah server mati .....	83
Gambar 5.34 Tampilan Web sebelum server dimatikan.....	84
Gambar 5.35 Tampilan server dimatikan .....	85
Gambar 5.36 Tampilan web ketika server dimatikan sebelum lewat 1 jam ..	85
Gambar 5.37 Tampilan server setelah lewat 1 jam.....	86
Gambar 5.38 Tampilan website setelah server dimatikan lewat 1 jam .....	86
Gambar 5.39 Tampilan website sebelum server dimatikan.....	87
Gambar 5.40 Tampilan server setelah dimatikan .....	88
Gambar 5.41 Tampilan website ketika server dimatikan sebelum lewat 1 jam .....	88
Gambar 5.42 Tampilan server setelah lewat 1 jam.....	89
Gambar 5.43 Tampilan website setelah server dimatikan lewat 1 jam .....	89

## DAFTAR TABEL

Tabel 2.1 Simbol dalam Use Case Diagram .....	16
Tabel 4.1 Tabel Perawatan .....	39
Tabel 4.2 Tabel Lokasi.....	39
Tabel 4.3 Tabel Detail jadwal .....	40
Tabel 4.4 Tabel Pengeluaran.....	40
Tabel 4.5 Tabel Izin Dokter .....	41
Tabel 4.6 Tabel Users .....	41
Tabel 4.7 Tabel Dokter .....	42
Tabel 4.8 Tabel Perawatan_reservasi.....	42
Tabel 4.9 Tabel Reservasi Rekam Medik .....	43
Tabel 4.10 Tabel Pasien .....	43
Tabel 4.10 Tabel Pasien(lanjutan).....	44
Tabel 5.1 Pengujian Aplikasi .....	69
Tabel 5.2 Pengujian kondisi benar input reservasi dari admin .....	71
Tabel 5.3 Pengujian alternatif reservasi dari admin.....	73
Tabel 5.4 Pengujian kondisi benar registrasi pasien .....	74
Tabel 5.5 Pengujian alternatif registrasi akun pasien.....	76
Tabel 5.6 Pengujian kondisi benar reservasi dari pasien .....	77
Tabel 5.7 Pengujian alternatif reservasi dari pasien.....	80
Tabel 5.8 Pengujian mode offline pada <i>website</i> .....	82
Tabel 5.9 Pengujian Sistem Membatasi akses ke cache jika lebih 1 jam .....	84
Tabel 5.10 Pengujian Sistem Lama waktu akses cache tidak dibatasi.....	87
Tabel 5.11 Kesimpulan hasil pengujian .....	90

## ABSTRAK

*Klinik Gigi Xenon Dental House adalah klinik gigi yang menyediakan layanan kesehatan gigi yang didirikan pada tahun 2020 di Kota Padang dan telah berkembang dengan memiliki cabang di Kota Bukittinggi. Dalam operasionalnya, klinik ini menghadapi beberapa tantangan dalam pengelolaan data dan layanan. Pertama, proses reservasi masih dilakukan secara manual melalui WhatsApp, yang dapat menyebabkan miskomunikasi dan kesalahan penjadwalan. Kedua, tidak adanya satu pusat data yang terintegrasi yang mana saat ini menggunakan berbagai media untuk mencatat data, mulai dari catatan manual hingga pencatatan reservasi melalui WhatsApp. Ketiga, pencatatan rekam medis masih menggunakan media kertas yang berisiko rusak atau hilang serta membutuhkan ruang penyimpanan fisik dan . Keempat, tidak adanya sistem visualisasi data real-time yang dapat menyulitkan pemantauan kinerja klinik, tren pasien, tren layanan perawatan, dan produktivitas dokter. Oleh karena itu, dibutuhkan sebuah sistem informasi manajemen klinik gigi untuk mengoptimalkan pengelolaan operasional klinik. Penelitian ini menggunakan metode waterfall dengan teknik pengumpulan data melalui observasi, wawancara, analisis dokumen, dan studi literatur. Sistem informasi yang dibangun berbasis web dengan mengimplementasikan teknologi Progressive Web App (PWA), menggunakan framework Laravel 11, dan API Twilio untuk integrasi notifikasi WhatsApp. Hasil penelitian menunjukkan bahwa sistem informasi manajemen klinik gigi berhasil dibangun sesuai dengan perancangan. Sistem ini menyediakan sentralisasi data, meningkatkan keamanan dan akurasi data, mengotomatisasi penjadwalan dan reservasi, serta menyediakan fitur visualisasi data untuk pemantauan kinerja klinik. Implementasi PWA memungkinkan akses offline, sementara integrasi WhatsApp memfasilitasi komunikasi antara klinik dan pasien.*

*Kata Kunci: Sistem Informasi Manajemen, Klinik Gigi, Progressive Web App, Laravel, Twilio, Visualisasi Data*

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Dalam era globalisasi yang semakin maju, perkembangan teknologi informasi menjadi esensial sebagai alat pendukung untuk mempermudah berbagai aktivitas di berbagai sektor, termasuk di dalamnya organisasi, lembaga, instansi, dan perusahaan (Kurniawan Ritonga and Firdaus, 2024). Penerapan teknologi informasi telah membawa berbagai kemudahan yang signifikan dalam kehidupan sehari-hari, termasuk di bidang perawatan kesehatan (Isana, Kumboyono and Windarwati, 2022). Meskipun sistem informasi telah ada dalam berbagai organisasi, termasuk klinik gigi, masih terdapat permasalahan mendasar yang perlu diatasi. (Indah *et al.*, 2021).

Sebagai contoh, kita bisa merujuk pada Klinik Gigi Xenon Dental House, yang merupakan salah satu penyedia layanan kesehatan gigi di Kota Padang. Setelah dilakukan wawancara dan karyawan dengan Pemilik Klinik Gigi Xenon Dental House, proses bisnis pada klinik ini masih melakukan sebagian besar administrasinya secara manual yang mana bisa dikembangkan menjadi proses bisnis yang lebih baik lagi. Pendaftaran pasien, dan proses pencatatan masih bergantung pada proses manual yang sangat bergantung dengan keterampilan dan ketelitian manusia, serta dapat menimbulkan *human error*.

Klinik Gigi Xenon Dental House melayani berbagai kelompok usia, mulai dari anak-anak hingga orang dewasa, yang dilayani oleh dokter-dokter dengan berbagai jadwal reservasi yang ada. Ketidaksesuaian jadwal reservasi terjadi akibat kesalahan dalam penjadwalan yang saat ini dicatat menggunakan aplikasi Whatsapp saja dan tidak menggunakan sistem khusus reservasi. Saat ini reservasi di klinik ini hanya dapat dilakukan dengan menghubungi nomor WhatsApp admin klinik atau datang langsung ke klinik. Selain itu, staf klinik harus membuat laporan rekam medis yang juga dilakukan secara manual menggunakan media kertas yang membutuhkan ruang fisik yang disimpan terpisah dan mudah rusak. Sistem manual yang ada saat ini juga tidak menyediakan kemampuan visualisasi *realtime* data yang mana menyebabkan kurangnya penyajian informasi yang jelas untuk memantau kinerja klinik, tren pasien, tren layanan perawatan, serta produktivitas dokter.

Sebagai contoh proses yang manual ini juga membuat ketidakefisienan pada proses klinik gigi Dental Echo Clinic, pengolahan data yang dilakukan secara manual membutuhkan waktu yang lama menimbulkan ketidak validan data dan kerahasiaan data tidak bisa terjaga dengan baik (Lestari, 2019).

Beberapa penelitian sebelumnya telah mengkaji sistem informasi manajemen dalam konteks pelayanan kesehatan. Sebagai contoh, penelitian oleh Mahdalena, Alamsyah & Sidik (2023) yang berjudul “Sistem Informasi Manajemen dan Keuangan Berbasis Web pada Klinik Gigi Eldental Banjarmasin”. Sistem ini bertujuan untuk meningkatkan kualitas pelayanan yang diberikan oleh klinik dengan mengatasi keterbatasan sistem manual yang ada saat ini untuk data pasien, pencatatan keuangan, dan rekam medis. Para peneliti menggunakan metode Waterfall dan bahasa pemrograman PHP dengan Framework Laravel untuk mengembangkan sistem, yang mencakup fitur-fitur seperti manajemen data untuk dokter, perawat, pasien, obat-obatan, keuangan, dan pelaporan. Web ini menghasilkan proses pendataan yang lebih efisien, dan mempermudah klinik dalam pembuatan laporan keuangan dan inventaris.

Penelitian oleh Rafiqah Majidah (Majidah et al., 2019) berjudul “Pengembangan Sistem Informasi Pengelolaan Klinik Gigi Berbasis *Website* Menggunakan Prinsip Point of Sale” juga membahas penerapan sistem informasi pada klinik. Jurnal mengusulkan sebuah sistem yang menggabungkan prinsip-prinsip point of sale dengan sistem berbasis *website* penerapan sistem informasi pengelolaan klinik gigi menggunakan *point of sale*. Selain itu, sistem ini juga membantu mengatasi masalah kurangnya efisiensi kerja dalam hal waktu dan upaya yang dikeluarkan hingga 30 kali lebih cepat dibandingkan dengan sebelum menggunakan sistem.

Penelitian lainnya yang berjudul "Sistem Informasi Manajemen Klinik Gigi Berbasis Client Server (Sari Ira Puspita , 2017). Penelitian ini membahas tentang sistem informasi manajemen klinik gigi yang bertujuan untuk menggantikan sistem pencatatan manual dan pengolahan data ke sistem komputerisasi, untuk mengatasi hambatan yang sering terjadi terkait dengan ketidakmampuan untuk menyediakan informasi secara cepat, akurat, dan tepat waktu. Sistem ini mencakup fitur-fitur seperti pendaftaran pasien, rekam medis, diagnosis, pengobatan, dan resep.

Penelitian ini menghasilkan peningkatan pelayanan pada pihak RSJ Pekanbaru serta memudahkan, memperpendek proses, dan menghemat waktu dalam pengerjaan proses pelayanan klinik gigi.

Penelitian-penelitian ini relevan dengan penelitian yang sedang dilakukan, yang bertujuan untuk mengembangkan sistem informasi manajemen klinik gigi yang lebih baik. Dari penelitian penelitian ini juga terlihat bahwa penerapan sistem informasi pada klinik dibutuhkan dan dapat meningkatkan kualitas dari pelayanan klinik. Meskipun terdapat penelitian terkait, belum ada yang menerapkan pendekatan *Progressive Web App (PWA)*. *PWA* merupakan teknologi Web-based development, yang memungkinkan agar suatu *website* dapat memiliki karakteristik / *experience* layaknya menggunakan suatu aplikasi mobile native, sehingga dapat memberikan pengalaman yang berkesan kepada user (Karli, Muawwal and Thayf, 2023). Dalam penelitian ini, sistem informasi yang dirancang akan menggunakan pendekatan *PWA* yang membuat *website* dapat dijalankan dalam keadaan *offline* dan lebih cepat dalam memuat data (Muddin, Tehuayo and Iksan, 2021). Sistem informasi ini semoga memberikan kemudahan kepada pasien dan membantu klinik dalam pendaftaran pasien, pengelolaan data yang, pelayanan yang lebih baik, peningkatan produktivitas, dan proses yang lebih baik. Sistem informasi ini diharapkan dapat diakses dan dimanfaatkan oleh masyarakat luas, sehingga dapat berfungsi secara optimal di Klinik Gigi Xenon Dental House.

Berdasarkan dari permasalahan tersebut dilaksanakan penelitian di Klinik Gigi Xenon Dental House dengan judul “Rancang Bangun Sistem Informasi Manajemen Klinik Gigi (Studi Kasus : Klinik Gigi Xenon Dental House)”. Dengan penelitian ini, diharapkan dapat memberikan kontribusi yang berarti terhadap klinik ini dan pada akhirnya meningkatkan kualitas layanan kesehatan gigi di masyarakat.

## **1.2 Rumusan Masalah**

Berdasarkan latar belakang yang telah disebutkan sebelumnya, rumusan masalah dalam penelitian ini adalah bagaimana membangun sistem informasi manajemen klinik gigi pada klinik gigi Xenon Dental House.

## **1.3 Batasan Masalah**

Berdasarkan rumusan masalah seperti yang dijelaskan sebelumnya, maka batasan masalah dalam tugas akhir ini adalah:

1. Sistem informasi ini mencakup pengelolaan data pasien, dokter, pengelolaan data rekam medis, pengelolaan reservasi, izin dokter, pengelolaan data layanan perawatan, visualisasi data klinik.
2. Sistem informasi manajemen pada klinik gigi xenon dental house dilakukan sampai tahap implementasi dan pengujian.
3. Proses perancangan dan pembangunan sistem informasi hanya ditujukan pada klinik gigi xenon dental house
4. Pengujian aplikasi hanya sebatas memeriksa ketersediaan fungsional dan kesesuaian dengan rancangan sistem yang diusulkan.
5. Penerapan fitur *PWA* hanya sebatas penerapan fitur akses offline untuk website

## **1.4 Tujuan Penelitian**

Tujuan dari penelitian ini adalah membangun sebuah sistem informasi manajemen klinik gigi pada Klinik Gigi Xenon Dental House untuk mengatasi kendala serta membantu pada manajemen klinik gigi seperti pengelolaan data pasien, dokter, pengelolaan data rekam medis, pengelolaan reservasi, izin dokter, pengelolaan data pelayanan perawatan, visualisasi data klinik di Klinik Gigi Xenon Dental House.

## **1.5 Sistematika Penulisan**

Sistematika penulisan penelitian ini adalah sebagai berikut:

## **BAB I : PENDAHULUAN**

Bab ini berisi tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, serta sistematika penulisan laporan.

## **BAB II : TINJAUAN PUSTAKA**

Bab ini berisi tentang landasan teori dan informasi pendukung yang digunakan untuk penelitian ini.

## **BAB III : METODOLOGI PENELITIAN**

Bab ini menjelaskan tentang objek penelitian, metode pengumpulan data, metode pengembangan sistem yang digunakan, dan flowchart penelitian.

## **BAB IV: ANALISIS DAN PERANCANGAN SISTEM**

Bab ini berisi tentang analisis sistem yang berjalan, analisis kebutuhan dan perancangan pada sistem usulan untuk menjawab permasalahan pada sistem lama yang digambarkan melalui diagram dan tools pendukung lainnya.

## **BAB V: IMPLEMENTASI DAN PENGUJIAN**

Bab ini berisi tentang implementasi dari hasil analisis dan perancangan sistem yang telah dijabarkan pada bab sebelumnya. Pada bab ini juga akan dijelaskan pengujian yang dilakukan terhadap sistem usulan berdasarkan rancangan yang telah dibuat pada bab sebelumnya.

## **BAB VI: PENUTUP**

Bab ini berisi tentang kesimpulan dari penelitian dan saran terhadap pengembangan kedepannya.



## **BAB II**

### **TINJAUAN PUSTAKA**

Bab ini menjelaskan teori dan informasi pendukung yang digunakan dalam penelitian pembangunan sistem informasi manajemen klinik gigi xenon dental house.

#### **2.1 Klinik Gigi**

Dalam Peraturan Menteri Kesehatan nomor 028/Menkes/Per/I/2011 (Kemenkes RI 2011, 2011), klinik dijelaskan sebagai fasilitas pelayanan kesehatan yang menyelenggarakan pelayanan kesehatan perorangan yang menyediakan pelayanan medis dasar dan atau spesialis, diselenggarakan oleh lebih dari satu jenis tenaga kesehatan dan dipimpin oleh seorang tenaga medis.. Lebih khusus, klinik gigi adalah klinik yang memberikan pelayanan medik dasar yang dapat mengkhususkan pelayanan pada satu bidang tertentu berdasarkan disiplin ilmu, organ atau jenis penyakit tertentu(Kemenkes RI 2011, 2011) .

Biasanya, ketika seseorang datang ke klinik atau praktek dokter gigi pribadi, tidak disebutkan secara eksplisit kategori atau jenis klinik gigi yang dikunjungi, karena klinik gigi yang dikunjungi cenderung menyediakan beragam jenis perawatan gigi dan mulut. Namun, dalam konteks rumah sakit atau poliklinik gigi, ruang perawatan gigi sering kali dibagi menjadi berbagai kategori sesuai dengan jenis perawatan yang disediakan oleh dokter gigi yang bertugas di klinik tersebut. Setiap jenis klinik gigi umumnya memiliki dokter gigi spesialis yang berkompeten di bidangnya(Muntihana *et al.*, 2017).

Klinik tidak hanya berfungsi sebagai tempat untuk memberikan layanan medis, tetapi juga memiliki peran penting dalam membantu masyarakat sekitarnya melalui pengamatan terhadap kondisi tubuh mereka (Tambunan and Malau, 2022). Pengamatan ini dapat dilakukan melalui pemeriksaan terhadap keluhan-keluhan yang disampaikan oleh pasien. Seluruh hasil pemeriksaan dokter kemudian terdokumentasikan dalam rekam medik, mencakup diagnosa penyakit, tindakan medis, dan resep obat (Yossiant and Hosizah, 2023). Proses di klinik, selain mencakup pemeriksaan kondisi tubuh pasien, juga melibatkan administrasi klinik yang berfungsi untuk mencatat semua kegiatan sesuai dengan proses bisnis yang

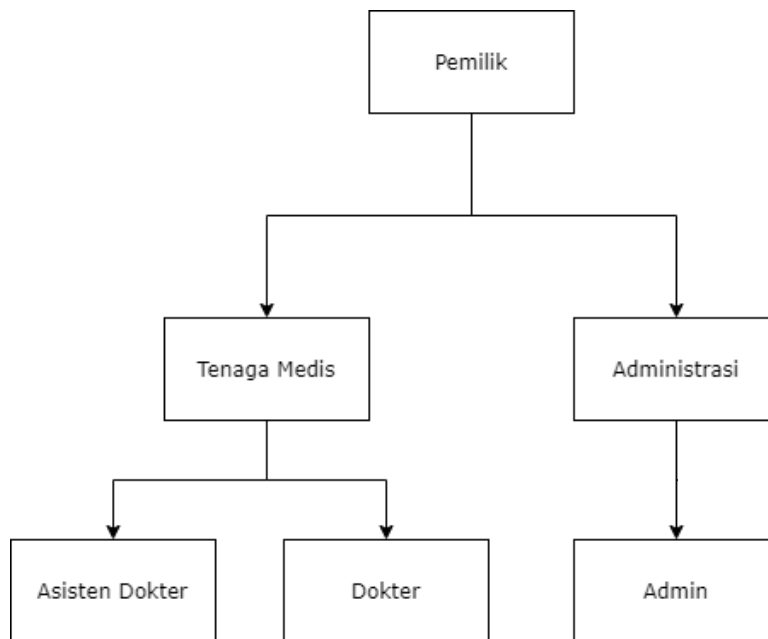
berlangsung (Saptiono Mulana, 2020). Hal ini mencakup pendaftaran pasien, pencatatan data obat, pemeriksaan laboratorium, dan pembayaran obat. Pentingnya informasi dalam konteks pelayanan kesehatan, termasuk di dalamnya klinik, sangat besar. Informasi yang diperoleh dengan cepat, tepat, dan akurat akan berkontribusi pada penyelenggaraan layanan yang terbaik bagi pasien, meningkatkan tingkat kepuasan mereka (Amalia and Huda, 2020). Penerapan sistem informasi dalam pengolahan data di klinik akan membantu proses tersebut.

Klinik, sebagai fasilitas pelayanan kesehatan, memiliki peran signifikan tidak hanya dalam merawat pasien, tetapi juga dalam membantu masyarakat sekitar melalui observasi terhadap kondisi tubuh mereka. Observasi ini dilakukan melalui pemeriksaan terhadap keluhan-keluhan pasien, di mana hasilnya dicatat dalam rekam medik yang mencakup diagnosa penyakit, tindakan dokter, dan resep obat (Cahyani *et al.*, 2024). Meskipun fokus utama adalah pada pemeriksaan kondisi kesehatan pasien, klinik juga melibatkan administrasi yang mendukung proses bisnisnya. Administrasi klinik mencakup kegiatan seperti pendaftaran pasien, pencatatan rekam medik, dan pengajuan pengambilan obat (Khoirunnisa *et al.*, 2021). Pentingnya informasi dalam pelayanan kesehatan seperti informasi yang cepat, tepat, dan valid akan memberikan kepuasan pasien.

Penting untuk diingat bahwa klinik gigi tidak hanya berperan sebagai penyedia layanan medis, tetapi juga sebagai entitas yang mendukung kebutuhan administratif dan informasional (Tambunan and Malau, 2022). Penggunaan teknologi, seperti pengolahan data melalui komputer, di klinik dapat membantu proses tersebut. Informasi yang diperoleh dengan cepat, tepat, dan akurat tidak hanya memperbaiki penyelenggaraan layanan bagi pasien, tetapi juga membantu dalam manajemen administratif, termasuk perekaman data pasien, pengelolaan inventaris obat, dan pelaporan hasil pemeriksaan laboratorium. Dengan demikian, klinik gigi bukan hanya tempat untuk merawat kesehatan gigi masyarakat, tetapi juga menjadi tempat untuk menjaga dan meningkatkan kesehatan serta kepuasan pasien (Nuryani, 2021).

## 2.2 Klinik Gigi Xenon Dental House

Klinik gigi Xenon Dental House, adalah klinik yang berlokasi di Jl. Palembang No 11, Ulak Karang, Kota Padang, Sumatera barat merupakan sebuah pusat pelayanan kesehatan gigi yang didirikan pada tahun 2020 melalui kolaborasi beberapa dokter gigi. Saat ini klinik gigi Xenon Dental House sudah terdapat di 2 kota yaitu kota padang dan kota bukittinggi. Dalam jalannya klinik ini tentunya terdapat elemen elemen yang mendukung klinik gigi Xenon Dental House sebagai organisasi, adapun struktur organisasi dari gigi Xenon Dental House dapat dilihat pada gambar 2.1 :



Gambar 2.1 Struktur Organisasi Klinik Gigi Xenon Dental House (2024)

Dari diagram diatas, pada klinik gigi Xenon Dental House terdapat beberapa peran yang bertanggung jawab. Pemilik bertanggung jawab atas manajemen keseluruhan dan pengambilan keputusan strategis. Dokter gigi adalah tenaga medis yang memberikan pelayanan kesehatan gigi kepada pasien, sementara asisten dokter mendukung tugas dokter dengan membantu dalam prosedur klinis. Administrasi klinik bertanggung jawab mengelola aspek administratif klinik, seperti pengelolaan jadwal, sosial media, reservasi. Sementara itu, resepsionis berperan sebagai penerima tamu, menjawab panggilan.

Klinik ini awalnya berfokus pada penyediaan peralatan dan keperluan perawatan gigi. Namun, seiring perkembangannya, Klinik gigi Xenon Dental House mulai menyediakan berbagai layanan kesehatan gigi. Proses pelayanan di klinik ini melibatkan langkah-langkah berikut: pasien melakukan reservasi dengan mengunjungi langsung atau menghubungi melalui aplikasi WhatsApp milik Klinik gigi Xenon Dental House. Data identitas pasien dan keluhan pasien dicatat, lalu pasien diberikan jadwal untuk tindakan dokter. Sehari sebelum jadwal tindakan, admin klinik mengirimkan pengingat kepada pasien. Saat pasien datang, tindakan medis dilakukan sesuai dengan penilaian dokter. Pasien diminta untuk menjadwalkan kunjungan kontrol jika diperlukan. Klinik ini menyediakan berbagai jenis tindakan, termasuk konsultasi dokter, tambal gigi, pencabutan gigi, scaling gigi, dan pemasangan behel.

### **2.3 Sistem Informasi Manajemen**

Manajemen sendiri mencakup proses perencanaan, pengorganisasian, pengawasan, pengarahan, dan lain-lain, dalam suatu organisasi. Sedangkan, informasi dalam satu organisasi adalah data yang diolah sedemikian rupa sehingga memiliki nilai dan arti bagi organisasi. Sistem Informasi Manajemen (SIM) merupakan sistem yang mengolah serta mengorganisasikan data dan informasi yang berguna untuk mendukung pelaksanaan tugas dalam suatu organisasi (Hariyanto, 2018).

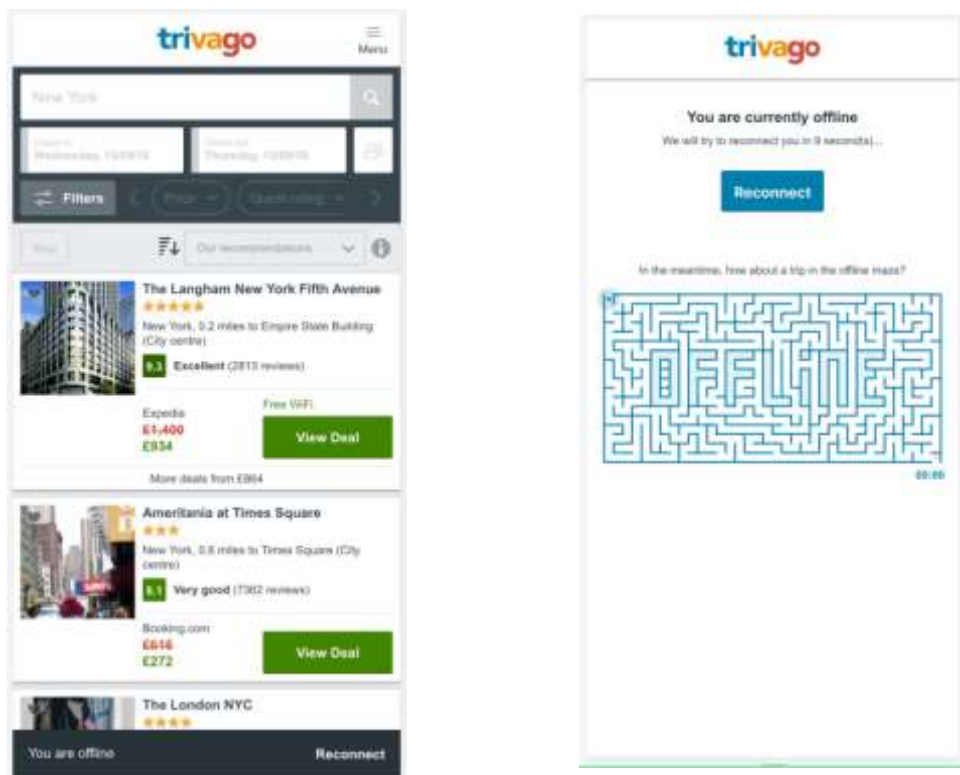
Sistem Informasi Manajemen dapat berupa sistem informasi berikut ini (Wijoyo, 2021) :

- 1) Sistem Informasi Akuntansi
- 2) Sistem Informasi Pemasaran
- 3) Sistem Informasi Manajemen Persediaan.
- 4) Sistem Informasi Personalia.
- 5) Sistem Informasi Distribusi.
- 6) Sistem Informasi Pembelian.
- 7) Sistem Informasi Analisa Kredit.
- 8) Sistem Informasi Analisa Software.
- 9) Sistem Informasi riset dan pengembangan.

- 10) Sistem Informasi kekayaan.
- 11) Sistem Informasi Teknis.

## 2.4 Progressive Web App (PWA)

*Progressive Web App* adalah suatu teknik bagaimana kita dapat mengakses dengan kencang dan cepat di *website* dan aplikasi, aplikasi bisa ditampilkan dalam bentuk *website* ataupun aplikasi. *PWA* adalah sebuah konsep pengembangan web yang bertujuan untuk memberikan pengalaman yang mirip dengan aplikasi mobile pada peramban (browser) desktop dan perangkat mobile. *PWA* memanfaatkan kemampuan peramban modern, seperti *Service Workers* dan *Web App Manifest*, untuk menciptakan aplikasi web yang lebih kuat dan dapat diakses secara offline (Haryanto and Saputra Elsi, 2021). Contoh penerapan *PWA* bisa dilihat pada gambar ini Gambar 2.2 berikut ini.



Gambar 2.2 Penerapan *PWA* pada *website* trivago(Warrender, 2018)

*PWA* memiliki beberapa manfaat seperti berikut (Aleksandrs Hodakovskis,2019) :

Sebagai pemilik situs web atau *e-commerce*:

- Memberikan pengalaman website yang responsif
- Performa lebih cepat
- Meningkatkan pengalaman pengguna
- Meningkatkan tingkat pencarian website
- PWA biasanya lebih terjangkau untuk dibuat dan dipelihara dibandingkan aplikasi native
- User dapat menyimpan aplikasi di layar utama mereka
- Semua hal di atas menghasilkan tingkat kinerja yang lebih tinggi secara keseluruhan keterlibatan, yang mengarah pada peningkatan pendapatan

Sebagai pengguna:

- Waktu muat lebih cepat dan instan
- Penjelajahan(searching) offline
- Pengalaman pencarian yang lebih baik dan lancar
- PWA menggunakan lebih sedikit data
- Akses sekali klik (bila disimpan ke layar beranda perangkat Anda)

## **2.5 Alat Analisis dan Perancangan**

Dalam bagian ini, diberikan penjelasan mengenai teori-teori terkait dengan *tools* analisis dan perancangan, seperti Business Process Modelling Notation (BPMN), Use Case, dan Entity Relationship Diagram (ERD). Alat-alat ini memiliki hubungan penting dalam proses analisis dan perancangan yang terjadi dalam konteks penelitian ini.

### **2.5.1 Business Process Modelling Notation (BPMN)**

*Business Process Modelling Notation* adalah sebuah standar yang digunakan untuk memodelkan proses bisnis dengan menyediakan notasi grafis dalam pemodelan proses bisnis tersebut. BPMN menjelaskan diagram proses bisnis yang disusun untuk membuat model grafis dari proses bisnis dengan aktivitas dan kontrol aliran yang mendefinisikan urutan kerja berdasarkan pendekatan diagram alur (Yohana and Marisa, 2018).

*Business Process Modelling Notation* terdiri atas empat kategori elemen (Ismanto, Hidayah and Charisma, 2020), yaitu sebagai berikut.

### 1. *Swimlanes*

Elemen ini digunakan untuk mengatur dan memisahkan peran atau tanggung jawab dari suatu proses. Notasi yang digunakan adalah *pool* dan *lane*. *Pool* adalah kontainer dari satu proses. Sedangkan *lane* adalah partisi dari suatu proses, yang menunjukkan sub organisasi, jabatan, peran atau penanggungjawab. Notasi *swimlanes* dapat digambarkan sesuai Gambar 2.3.



Gambar 2. 3 Notasi Swimlanes (Ismanto, Hidayah and Charisma, 2020)

### 2. *Connecting Object*

Elemen ini digunakan untuk menggambarkan aliran pesan antar proses dimana satu kejadian dengan kejadian yang lain saling berhubungan dan merepresentasikan dari hubungan tersebut. Terdapat tiga notasi yang digunakan pada *connecting object*:

- Sequence flow*, konektor yang menghubungkan antar objek yang mengalir dalam satu proses (*pool*).
- Message flow*, konektor yang menghubungkan antar objek yang mengalir antar proses (beda *pool*).
- Association*, konektor yang menghubungkan objek yang mengalir ke *artifact*.

Masing-masing notasi *connecting object* dapat digambarkan seperti Gambar 2.4.

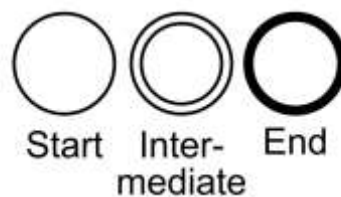


Gambar 2. 4 Notasi Connecting Object (Ismanto, Hidayah and Charisma, 2020)

### 3. Flow Object

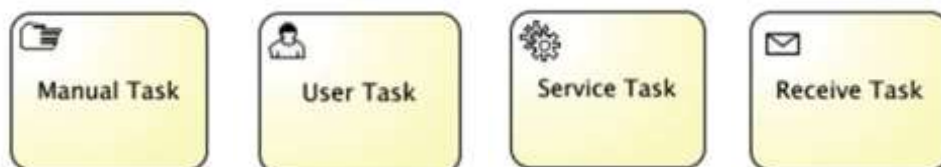
Elemen ini digunakan untuk menunjukkan objek yang mengalir pada suatu proses. Terdapat tiga notasi yang digunakan pada *flow object*:

a. *Event*, menjelaskan apa yang terjadi saat itu. *Event-event* ini mempengaruhi alur proses dan biasanya menyebabkan terjadinya kejadian (*trigger*) atau sebuah dampak (*result*). Ada tiga macam *event*, yaitu *start event*, *intermediate event*, dan *end event* yang dapat dinotasikan sesuai Gambar 2.5.



Gambar 2. 5 Notasi Start, Intermediate, dan End Event(Wagner, 2017)

b. *Activity*, merepresentasikan pekerjaan (*task*) dan subproses (serangkaian pekerjaan) yang harus diselesaikan. Pekerjaan memiliki beberapa kategorikan seperti pada Gambar 2.6 berikut.



Gambar 2. 6 Manual, user, service, receive task (Stiehl, Raw and Smith, 2014)

c. *Gateway*, digunakan untuk mengontrol divergensi dan konvergensi *sequential flow*. Notasi *Gateway* dapat dilihat pada Gambar 2.7.



Gambar 2. 7 Notasi Gateway(Stiehl, Raw and Smith, 2014)

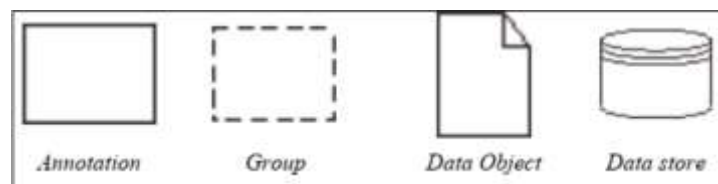


#### 4. *Artifacts*

Elemen yang digunakan untuk informasi tambahan pada suatu proses. Terdapat empat notasi yang digunakan pada *artifacts*:

- a. *Annotation*, penjelasan dari suatu objek yang mengalir.
- b. *Group*, pengelompokan dari beberapa objek yang mengalir.
- c. *Data object*, file dan dokumen yang digunakan dan dihasilkan oleh suatu aktifitas.
- d. *Data store*, sistem dan aplikasi yang digunakan dan dihasilkan oleh suatu aktifitas.

Notasi *artifacts* dapat digambarkan seperti Gambar 2.8.








Gambar 2. 8 Notasi Artifacts (Ismanto, Hidayah and Charisma, 2020)

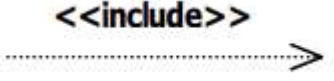
#### 2.5.2 Use Case Diagram

Diagram Use Case adalah representasi visual yang menggambarkan fungsi yang diinginkan dari suatu sistem, dengan penekanan pada apa yang dilakukan oleh sistem, bukan bagaimana melakukannya. Setiap use case menggambarkan interaksi antara aktor (pengguna atau elemen luar) dengan sistem. Penggunaan Diagram Use Case bermanfaat dalam merinci kebutuhan sistem, berkomunikasi dengan klien mengenai desain sistem, dan merencanakan uji coba untuk semua fitur system (Dharwiyanti, 2003). Simbol-simbol yang digunakan dalam Diagram Use Case dapat ditemukan dalam Tabel 2.1 berikut ini (Astuti, 2009).

Tabel 2.1 Simbol dalam Use Case Diagram

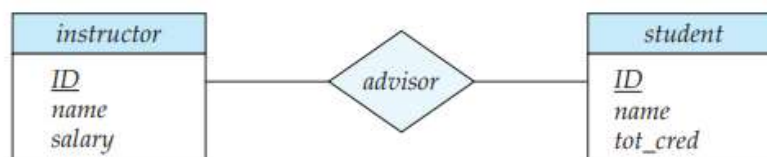
No	Gambar	Nama	Keterangan
1		Use Case	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase nama use case
2		Aktor	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri
3		Asosiasi	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case memiliki interaksi dengan aktor
4		Ekstensi	Relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan dapat berdiri sendiri walau tanpa use case tambahan
5		Generalisasi	Hubungan generalisasi dan spesialisasi (umum - khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya

Tabel 2.1 Simbol dalam *Use Case Diagram* (lanjutan)

6		Include	<p>relasi use case tambahan ke sebuah use case di mana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan use case ini</p>
---	---	---------	--

### 2.5.3 Entity Relationship Diagram(ERD)

Menurut Pulungan (Pulungan *et al.*, 2023) menyatakan bahwa ERD merupakan salah satu diagram utama representasi model data konseptual yang mencerminkan persyaratan data pengguna dalam sistem basis data.. Menurut ('Afiifah, Azzahra and Anggoro, 2022), ERD ini memrepresentasikan bagaimana entitas saling terkait antara satu dengan yang lainnnya dalam database. ERD digunakan untuk pemodelan basis data relasional. ERD dibentuk dari beberapa komponen yang saling berhubungan, dapat dilihat pada gambar 2.9 yang menggambarkan ERD antara instruktur dan pelajar (Silberschatz, 2011)



Gambar 2.9 ERD hubungan instruktur dan pelajar (Silberschatz, 2011)

ERD pada gambar 2.9 diatas terdiri dari beberapa komponen sebagai berikut

#### 1) Entitas :

Entitas adalah "benda" atau "objek" di dunia nyata yang dapat dibedakan dari objek lainnya. Misalnya, setiap orang di universitas adalah entitas.

2) Relasi :

Relasi adalah asosiasi di antara beberapa entitas. Sebagai contoh, kita bisa mendefinisikan relasi "*advisor*" yang mengasosiasikan instruktur A dengan mahasiswa universitas A, yang menyatakan bahwa instruktur A adalah pembimbing mahasiswa A

3) Atribut :

Atribut adalah properti atau karakteristik dari suatu entitas. Setiap atribut memiliki sekumpulan nilai yang disebut domain. Contoh seseorang memiliki atribut nama

## **BAB III**

### **METODE PENELITIAN**

Bab ini menjelaskan tentang objek penelitian, metode pengumpulan data, dan flowchart penelitian pada pengembangan sistem informasi manajemen klinik gigi Xenon Dental House.

#### **3.1 Objek Penelitian**

Klinik gigi Xenon Dental House, yang berlokasi di Jl. Palembang No 11, Ulak Karang, Kota Padang, Provinsi Sumatra Barat merupakan pusat pelayanan kesehatan gigi yang pada awalnya didirikan pada tahun 2020 melalui kolaborasi beberapa mahasiswa kedokteran gigi. Klinik ini awalnya berfokus pada penyediaan peralatan dan keperluan perawatan gigi, namun seiring perkembangannya, telah berkembang menjadi penyedia berbagai layanan kesehatan gigi. Proses pelayanan di klinik ini dimulai dengan reservasi oleh pasien melalui kunjungan langsung atau melalui aplikasi WhatsApp milik Klinik gigi Xenon Dental House. Data identitas dan keluhan pasien dicatat, dan pasien diberikan jadwal untuk tindakan dokter. Sehari sebelum jadwal tindakan, admin klinik mengirimkan pengingat kepada pasien. Saat pasien datang, tindakan medis dilakukan sesuai penilaian dokter, dan pasien dijadwalkan untuk kunjungan kontrol jika diperlukan. Jenis tindakan yang disediakan oleh klinik meliputi konsultasi dokter, tambal gigi, pencabutan gigi, scaling gigi, dan pemasangan behel.

Klinik ini awalnya berfokus pada penyediaan peralatan dan keperluan perawatan gigi. Namun, seiring perkembangannya, Klinik gigi Xenon Dental House mulai menyediakan berbagai layanan kesehatan gigi dan telah berkembang pesat dengan kini memiliki cabang di dua kota, yaitu Kota Padang dan Kota Bukittinggi. Kolaborasi antara beberapa dokter gigi membentuk dasar keberhasilan klinik ini sebagai sebuah organisasi. Fokus awal klinik terhadap peralatan dan perawatan gigi telah berkembang seiring waktu, mengakomodasi berbagai layanan kesehatan gigi. Proses pelayanan yang melibatkan reservasi, pencatatan data pasien, penjadwalan, dan tindakan medis sesuai dengan penilaian. Jenis layanan yang ada seperti konsultasi dokter, tambal gigi, pencabutan gigi, scaling gigi, dan pemasangan behel.

## **3.2 Metode Pengumpulan Data**

Dalam proses pengumpulan data untuk pengembangan aplikasi ini, metode yang digunakan mencakup studi literatur dan studi lapangan. Studi lapangan melibatkan observasi, wawancara, dan analisis dokumen.

### **3.2.1 Observasi**

Pengumpulan data dimulai dengan observasi, yang memungkinkan peneliti untuk memeriksa secara langsung operasi bisnis yang terjadi di Xenon Dental House. Dalam tahap observasi ini, proses bisnis yang diobservasi meliputi reservasi pasien, pengelolaan data pasien, serta aspek keuangan klinik diamati secara teliti.

### **3.2.2 Wawancara**

Pendekatan wawancara digunakan untuk menggali pemahaman yang lebih dalam mengenai proses bisnis yang sedang berlangsung di Xenon Dental House. Wawancara dengan pihak terkait di klinik bertujuan untuk mendapatkan informasi yang lebih rinci mengenai penelitian yang dilakukan. Wawancara dilakukan dengan memberikan pertanyaan kepada pemilik dari Xenon Dental House terkait kendala dan kebutuhan pada proses bisnis Xenon Dental House. Wawancara dilakukan untuk mengetahui bagaimana alur proses pengelolaan data pasien, dokter, keuangan, pengelolaan data rekam medis, pengelolaan jadwal rawat pasien, dan pembayaran

### **3.2.3 Analisis Dokumen**

Melalui analisis dokumen, data diperoleh dengan memeriksa dan mengevaluasi berbagai dokumen seperti dokumen data pasien, pencatatan penjadwalan reservasi pasien serta keuangan yang telah terkumpul dari objek penelitian. Dokumen-dokumen ini berkaitan dengan sistem informasi manajemen klinik gigi di Xenon Dental House, seperti catatan pasien, laporan keuangan, dan prosedur internal klinik.

### **3.2.4 Studi Literatur**

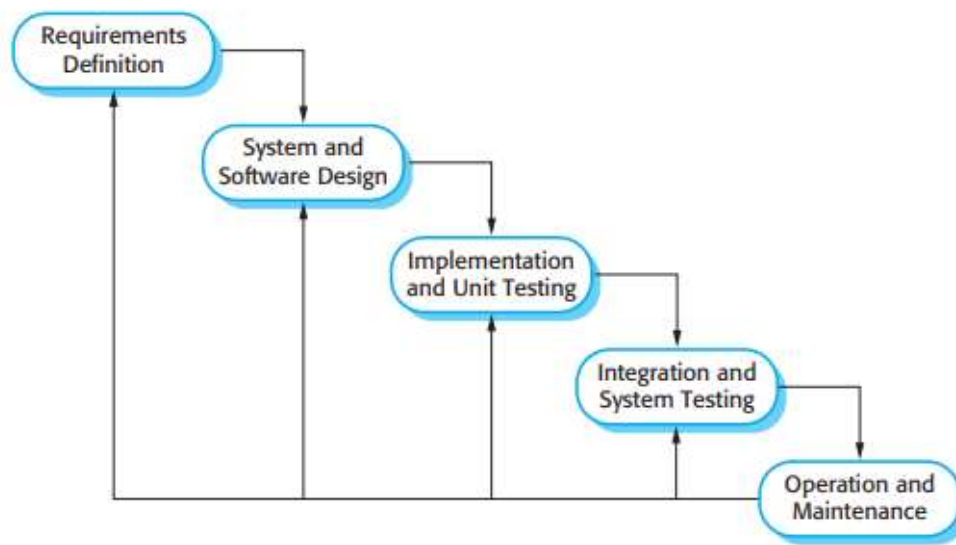
Studi literatur mencakup pengumpulan dan telaah berbagai sumber literatur yang relevan dengan penelitian ini. Informasi yang diambil dari literatur diperoleh

dari berbagai sumber, jurnal, buku, jurnal ilmiah, penelitian sebelumnya termasuk situs internet yang berkaitan dengan manajemen klinik gigi.

Pendekatan yang komprehensif ini memastikan bahwa data yang diperoleh dalam penelitian ini adalah sesuai dengan kebutuhan analisis dan pengembangan aplikasi, serta berdasarkan pemahaman mendalam tentang praktik bisnis di Xenon Dental House dan pengetahuan yang ada dalam literatur ilmiah.

### **3.3 Metode Pengembangan**

Metode yang digunakan dalam pengembangan aplikasi ini mengikuti pendekatan waterfall. Model pengembangan ini bersifat linear dari tahap awal pengembangan sistem yaitu tahap perencanaan sampai tahap akhir pengembangan sistem yaitu tahap pemeliharaan. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan dan tidak bisa kembali atau mengulang ke tahap sebelumnya (Pricillia and Zulfachmi, 2021). Kelebihan yang terkait dengan metode waterfall ini adalah Tahapan proses pengembangannya tetap (pasti), mudah diaplikasikan, dan prosesnya teratur dan cocok digunakan untuk produk software/program yang sudah jelas kebutuhannya. Namun, metode waterfall juga memiliki kelemahan, yaitu Terjadinya pembagian proyek menjadi tahap tahap yang tidak fleksibel, karena komitmen harus dilakukan pada tahap awal proses (Pricillia and Zulfachmi, 2021). Fase-fase pada pelaksanaan metode waterfall dapat dilihat pada Gambar 3.1 :



Gambar 3.1 Waterfall Model(Sommerville, 2011)

Dalam konteks pengembangan sistem informasi manajemen klinik gigi, proses hanya mencapai tahap keempat, yaitu tahap integrasi dan pengujian sistem. Hal ini disebabkan karena beberapa faktor, termasuk fokus pengerjaan tugas akhir ini adalah perancangan, pengembangan sistem, keterbatasan waktu yang ketat dan sumber daya yang terbatas jika dibandingkan dengan proyek komersial. Penjelasan untuk setiap tahap adalah sebagai berikut (Sommerville, 2011):

1. Analisis Kebutuhan dan Pendefinisian (Requirement Analysis and Definition)

Pada tahap ini, dilakukan analisis dan pengumpulan data untuk mengidentifikasi kebutuhan sistem berdasarkan data yang telah dikumpulkan. Pendekatan ini melibatkan analisis dokumen, wawancara, observasi, dan studi literatur yang relevan dengan pengembangan sistem informasi manajemen klinik gigi di Xenon Dental House. Hasil yang didapatkan dari tahap ini adalah BPMN, daftar fungsional pengguna, *use case diagram* dan *sequence diagram*.

2. Perancangan Sistem dan Perangkat Lunak (System and Software Design)

Tahap ini berkaitan dengan perencanaan solusi perangkat lunak. Data yang diperoleh selama tahap analisis digunakan untuk merancang sistem baru dengan mempertimbangkan struktur data, arsitektur perangkat lunak, perancangan basis



data, dan antarmuka aplikasi. Hasil yang didapatkan dari tahap ini adalah rancangan basis data, arsitektur aplikasi, class diagram dan mockup aplikasi.

### 3. Implementasi (Implementation)

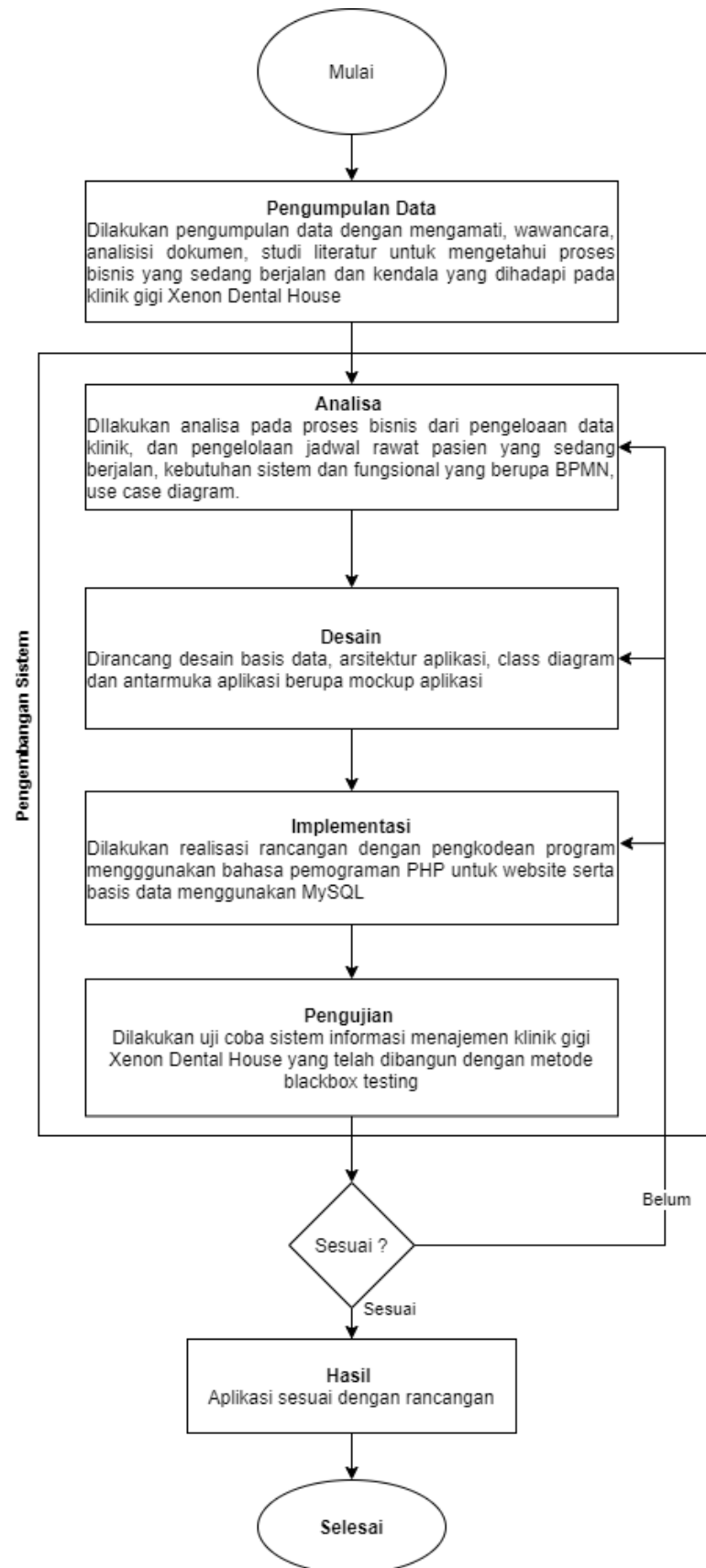
Tahapan ini adalah tahapan pembuatan program dan database dari design program dan design database yang sudah dibuat di tahap sebelumnya. Setiap modul program yang sudah dibuat akan diuji untuk menguji secara fungsionalitasnya. Dari hasil perancangan sistem, *requirement* perangkat lunak diubah menjadi kode pemrograman dalam aplikasi web menggunakan bahasa pemrograman PHP framework Laravel serta menerapkan pendekatan *Progressive Web App (PWA)*. Basis data yang digunakan dalam pembuatan web ini adalah MySQL.

### 4. Integrasi dan Pengujian sistem ( Integration dan System testing)

Tahap ini adalah tahapan unit-unit program atau program-program individual diintegrasikan dan diuji sebagai suatu sistem yang lengkap untuk memastikan bahwa persyaratan perangkat lunak telah terpenuhi. Pengujian sistem harus difokuskan pada pengujian interaksi komponen.

## 3.4 Flowchart Penelitian

Berdasarkan metode pengembangan sistem disusun sebuah flowchart atau diagram alur proses penelitian yang menggambarkan tahapan dalam pembangunan aplikasi manajemen laporan kriminal. Tahap-tahap dalam penelitian ini diilustrasikan dalam gambar 3.2 dibawah ini.



Gambar 3.2 Tahapan Penelitian

## **BAB IV**

### **ANALISIS DAN PERANCANGAN**

Bab ini membahas hasil analisis yang dilakukan terhadap kebutuhan perancangan sistem informasi manajemen klinik gigi berbasis web di klinik gigi Xenon Dental House. Analisis tersebut melibatkan penggunaan BPMN (Business Process Model and Notation), daftar fungsional pengguna, use case diagram dan sequence diagram. Sementara dalam fase perancangan sistem, digunakan entity relationship diagram (ERD), struktur database, dan antarmuka pengguna.

#### **4.1 Analisis Sistem**

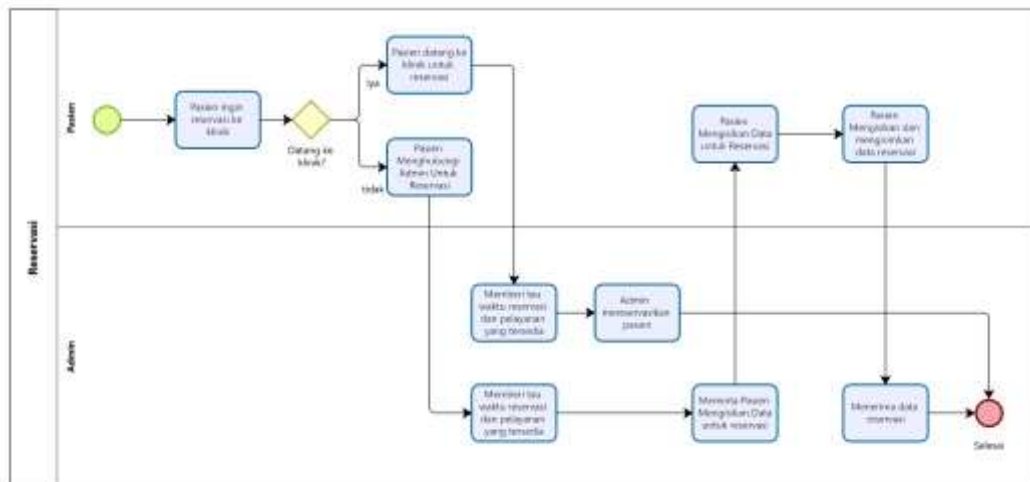
Analisis sistem merupakan proses memeriksa sistem yang sedang berjalan dan merancang sistem yang diusulkan agar memenuhi kebutuhan fungsional yang diinginkan dari sistem yang akan dibangun. Analisis sistem ini disusun dengan menggunakan BPMN, dan UML (Unified Modeling Language). UML yang diterapkan dalam analisis sistem ini meliputi sequence diagram dan use case diagram.

##### **4.1.1 Analisis sistem yang Sedang Berjalan**

Sistem yang sedang beroperasi ini diperoleh dari hasil pengumpulan data dengan cara analisis dokumen, wawancara, observasi dan analisis yang telah dilakukan di klinik gigi Xenon Dental House. Penggambaran atau pemodelan sistem yang berjalan akan disajikan menggunakan Business Process Model Notation terdiri dari sistem reservasi dan pelayanan, sistem manajemen jadwal.

###### **4.1.1.1 Sistem Reservasi**

Sistem reservasi pada klinik gigi xenon dental house dapat dilihat dalam Gambar 4.1.



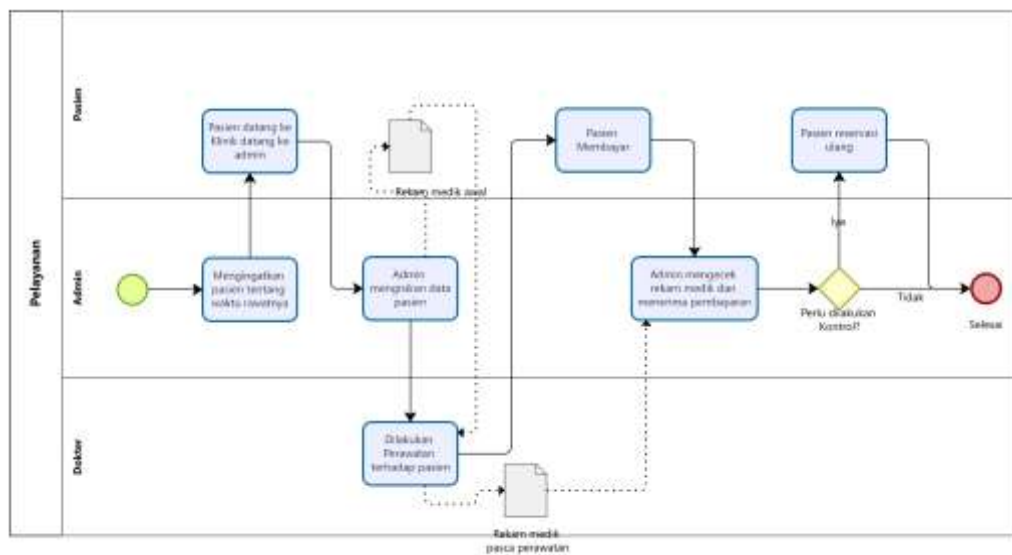
Gambar 4.1 BPMN Sistem reservasi

Berikut ini adalah proses bisnis reservasi yang sedang berjalan pada klinik gigi Xenon Dental House berdasarkan gambar 4.1.

1. Pasien bisa reservasi dengan cara datang reservasi langsung ke klinik gigi atau reservasi dengan menghubungi admin melalui aplikasi WhatsApp
2. Jika melalui whatsapp admin memberitahu waktu dan pelayanan yang tersedia serta meminta pasien untuk mengisi data untuk reservasi
3. Setelah pasien mengisi data reservasi dan memberikannya kepada admin, pasien akan menunggu sesuai waktu yang sudah direservasi.
4. Jika pasien datang ke klinik, maka admin akan langsung membantu mereservasi jadwal pasien
5. Proses selesai

#### 4.1.1.2 Sistem Pelayanan

Sistem pelayanan pada klinik gigi xenon dental house dapat dilihat dalam Gambar 4.2.



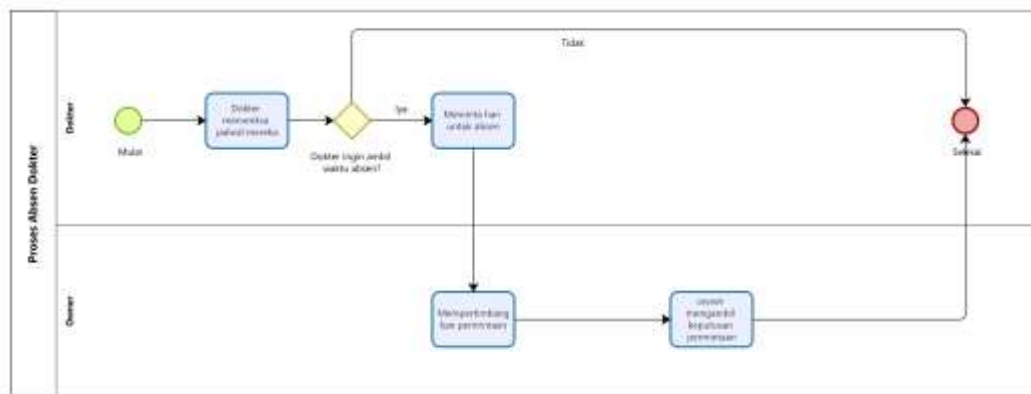
Gambar 4.2 BPMN Sistem pelayanan

Berikut ini adalah proses bisnis pelayanan yang sedang berjalan pada klinik gigi Xenon Dental House berdasarkan gambar 4.2.

1. Sebelum pasien dirawat, admin akan mengingatkan pasien untuk datang ke klinik untuk dilakukan perawatan yang sudah direservasi.
2. Pasien datang ke klinik dan diminta melakukan pendataan Kesehatan sebelum dilakukannya perawatan
3. Dokter melakukan perawatan terhadap pasien sesuai dengan kondisi pasien serta menuliskan kondisi pasien pada rekam medis.
4. Setelah perawatan selesai, pasien melakukan pembayaran
5. Jika dibutuhkan kontrol setelah perawatan maka pasien akan melakukan reservasi ulang kepada admin
6. Jika tidak diperlukan kontrol maka bisnis proses selesai

#### 4.1.1.3 Sistem perizinan dokter

Sistem perizinan dokter pada klinik gigi Xenon Dental House saat ini dapat dilihat dalam Gambar 4.3.



Gambar 4.3 BPMN manajemen jadwal

Berikut ini adalah proses bisnis perizinan dokter yang sedang berjalan pada klinik gigi Xenon Dental House berdasarkan gambar 4.3.

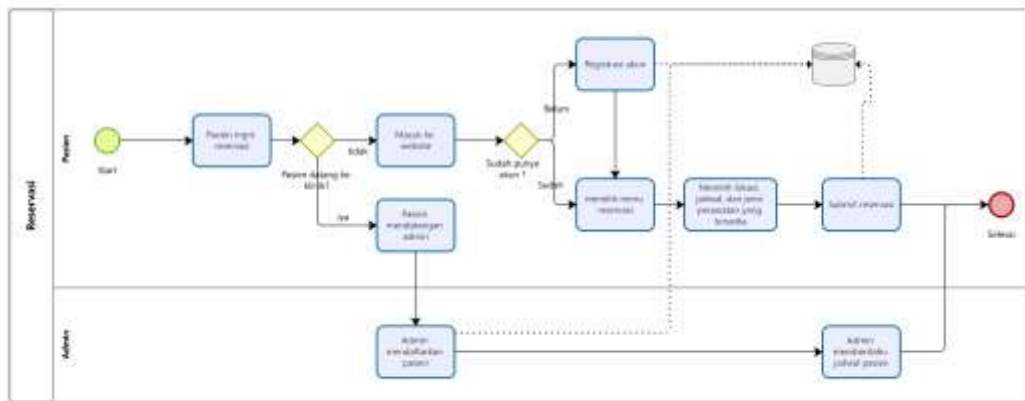
1. Dokter memeriksa jadwal mereka
2. Jika ingin mengambil absen, dosen akan meminta izin untuk absen kepada owner dari klinik di hari tertentu
3. Jika diizinkan maka dokter dibolehkan libur pada hari tersebut
4. Jika tidak diizinkan maka dokter tidak mendapatkan libur
5. Proses bisnis selesai

#### 4.1.2 Sistem yang Diusulkan

Dari sistem yang sudah berjalan, diusulkan sistem baru dengan membangun sistem informasi manajemen klinik gigi yang berbasis kan *website* pada klinik gigi Xenon Dental House. Sistem ini dimodelkan dengan BPMN, terdiri dari sistem reservasi dan pelayanan, sistem perizinan dokter.

##### 4.1.2.1 Sistem Reservasi

System reservasi dan pelayanan yang diusulkan dapat dilihat seperti pada gambar 4.4



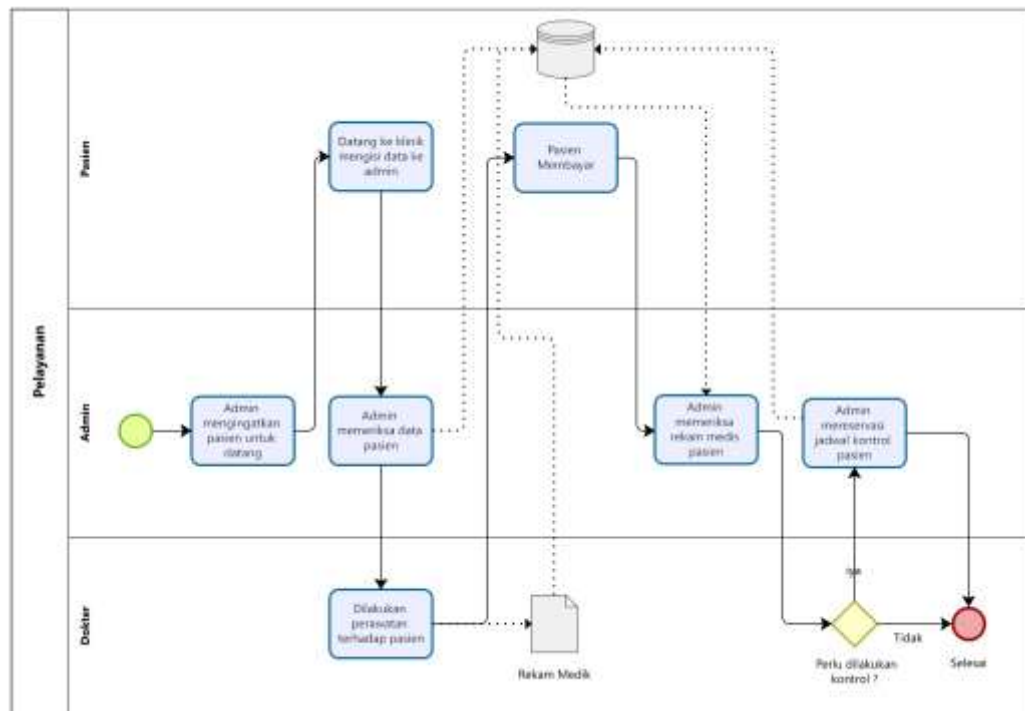
Gambar 4.4 BPMN Sistem reservasi

Berikut ini adalah proses bisnis reservasi dan pelayanan yang diusulkan pada klinik gigi Xenon Dental House berdasarkan gambar 4.4.

1. Pasien bisa reservasi dengan cara datang reservasi langsung ke klinik gigi atau reservasi dengan menggunakan *website* klinik gigi Xenon Dental House
2. Jika pasien datang ke klinik, maka pasien akan melakukan reservasi pada admin sesuai dengan jadwal yang tersedia melalui akun admin
3. Jika melalui *website*, maka pasien perlu membuka *website* klinik gigi untuk melakukan reservasi terlebih dahulu
4. Jika sudah memiliki akun pada *website* maka pasien bisa login di *website* klinik, jika belum registrasi akun terlebih dahulu dan melakukan login setelahnya
5. Pasien melakukan reservasi dengan memilih lokasi, jadwal dan jenis perawatan yang tersedia pada klinik gigi
6. Jika sudah, pasien mensubmit reservasi
7. Proses selesai

#### 4.1.2.2 Sistem Pelayanan

System pelayanan yang diusulkan pada klinik xenon dental house dapat dilihat seperti pada gambar 4.5



Gambar 4.5 BPMN Sistem pelayanan yang diusulkan

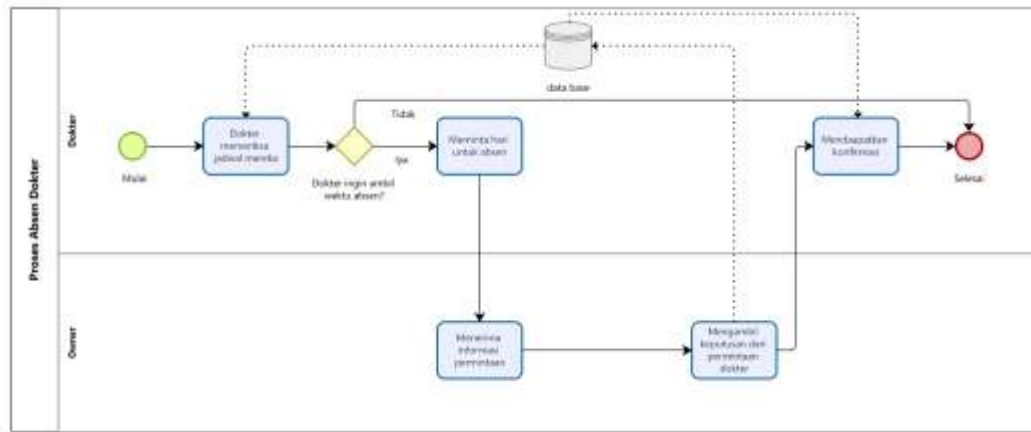
Berikut ini adalah proses bisnis reservasi dan pelayanan yang diusulkan pada klinik gigi Xenon Dental House berdasarkan gambar 4.3.

1. Sebelum pasien dirawat, admin akan mengingatkan pasien untuk datang ke klinik untuk dilakukan perawatan yang sudah direservasi.
2. Pasien datang ke klinik dan admin memeriksa data pasien
3. Dokter melakukan perawatan yang diperlukan terhadap pasien dan dokter menambahkan data rekam medis sesuai dengan kondisi pasien
4. Setelah selesai, pasien membayar
5. Admin memeriksa rekam medis pasien dan menerima pembayaran pasien
6. Jika diperlukan, admin mereservasi ulang untuk jadwal pasien
7. Jika tidak diperlukan maka proses bisnis selesai



#### 4.1.2.3 Sistem perizinan dokter

Sistem perizinan dokter pada klinik gigi Xenon Dental House yang diusulkan dapat dilihat dalam Gambar 4.6



Gambar 4.6 BPMN Perizinan dokter

Berikut ini adalah proses bisnis perizinan dokter yang diusulkan pada klinik gigi Xenon Dental House berdasarkan gambar 4.6.

1. Dokter login ke *website*
2. Dokter memeriksa jadwal mereka pada *website*
3. Jika ingin mengambil absen, dosen akan meminta izin untuk absen kepada owner dari klinik di hari tertentu
4. Jika diizinkan maka dokter dibolehkan libur dan dokter menginputkan data tersebut pada *website*
5. Jika tidak diizinkan maka dokter tidak mendapatkan libur
6. Proses bisnis selesai

#### 4.1.3 Analisis Kebutuhan Fungsional

Dari analisis alur proses *website* sistem informasi manajemen klinik gigi yang diajukan, beberapa kebutuhan fungsional dapat dirumuskan, antara lain:

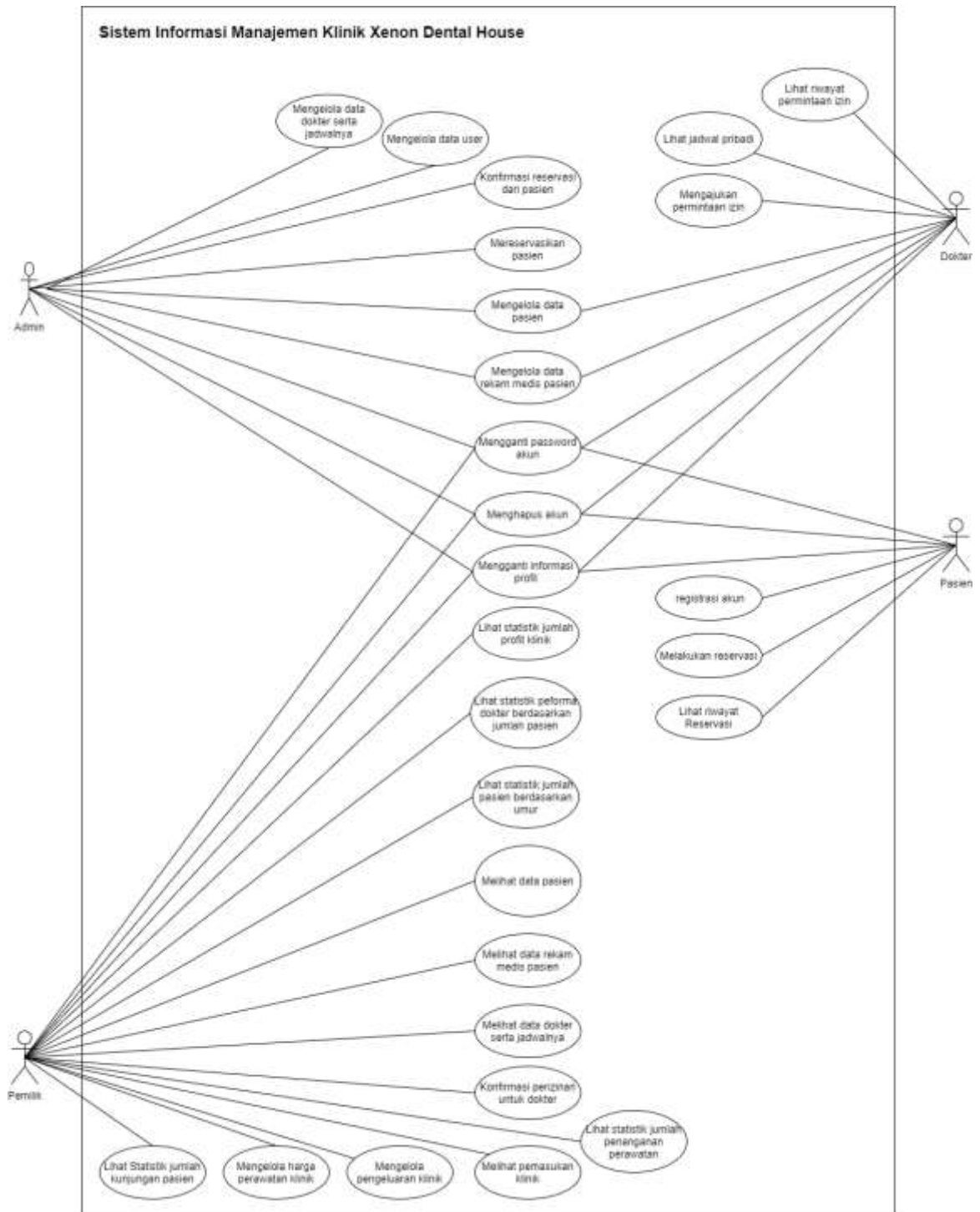
1. User dapat login pada *website*
2. User dapat mengganti password akun mereka

3. User dapat mengganti informasi profil mereka
4. User dapat menghapus akun mereka
5. Admin dapat melihat dashboard
6. Admin dapat mengkonfirmasi reservasi dari pasien
7. Admin dapat membantu reservasi pasien di klinik
8. Admin dapat mengelola data pasien
9. Admin dapat mengelola data rekam medis pasien
10. Admin dapat mengelola data dokter serta jadwal dokter
11. Admin dapat mengelola data data user
12. Pasien dapat registrasi akun
13. Pasien dapat melakukan reservasi perawatan
14. Pasien dapat melihat riwayat reservasi
15. Dokter dapat melihat jadwal mereka
16. Dokter dapat mengelola data pasien
17. Dokter membuat permintaan izin pada hari tertentu
18. Dokter dapat melihat permintaan izin mereka
19. Pemilik dapat melihat jumlah kunjungan pasien
20. Pemilik dapat melihat jumlah profit klinik
21. Pemilik dapat melihat jumlah penanganan perawatan
22. Pemilik dapat melihat performa dokter berdasarkan jumlah pasien
23. Pemilik bisa melihat data pasien berdasarkan umurnya
24. Pemilik bisa melihat data dokter
25. Pemilik bisa melihat data pasien

- 26. Pemilik bisa melihat data pemasukan
- 27. Pemilik bisa memutuskan perizinan dokter
- 28. Pemilik dapat mengelola pencatatan pengeluaran klinik
- 29. Pemilik dapat mengelola harga perawatan

#### **4.1.4 Use Case Diagram**

Berdasarkan analisis kebutuhan fungsional yang telah dilakukan pada sub bab sebelumnya, setiap kebutuhan fungsional tersebut dihubungkan dengan aktor yang terlibat dalam sistem, yang kemudian dimodelkan menjadi use case diagram. Setiap aktor memiliki hak akses terhadap fungsionalitas sistem yang berbeda sesuai dengan peran dan tanggung jawabnya masing-masing. Semua fungsionalitas ini bisa diakses setelah user melakukan login. Use case diagram ini dapat dilihat pada Gambar 4.7 di bawah ini.



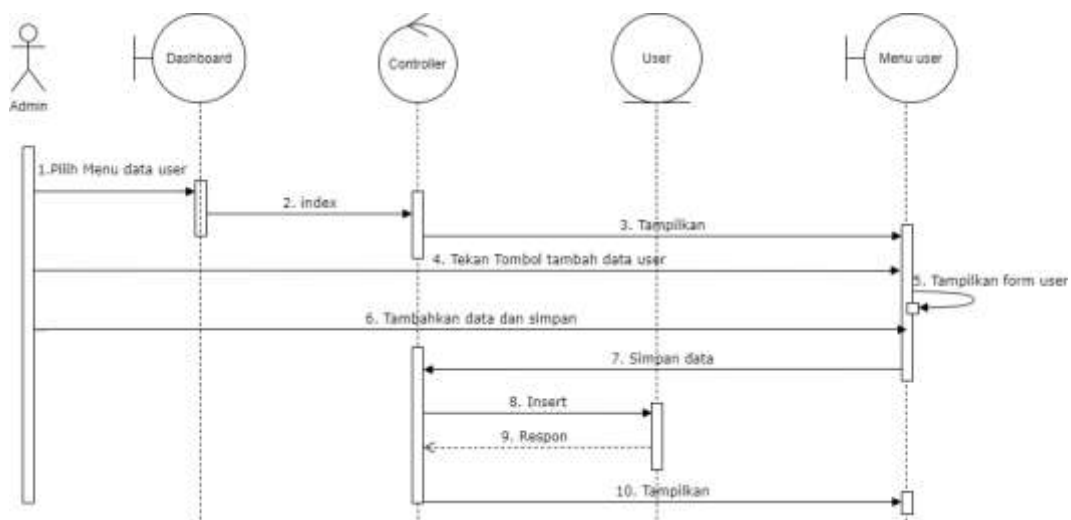
Gambar 4.7 Use Case Diagram Sistem

#### 4.1.5 Sequence Diagram

Sequence diagram menggambarkan aliran pesan dan data dalam proses interaksi antar objek yang terlibat dalam sistem secara berurutan. Sequence diagram yang akan dibahas di subbab ini mencakup proses login user, reservasi, proses menambahkan data user, dashboard owner . Sequence diagram lainnya dapat ditemukan pada Lampiran B.

#### 5.2.2.4 Sequence Diagram Menginputkan Data User

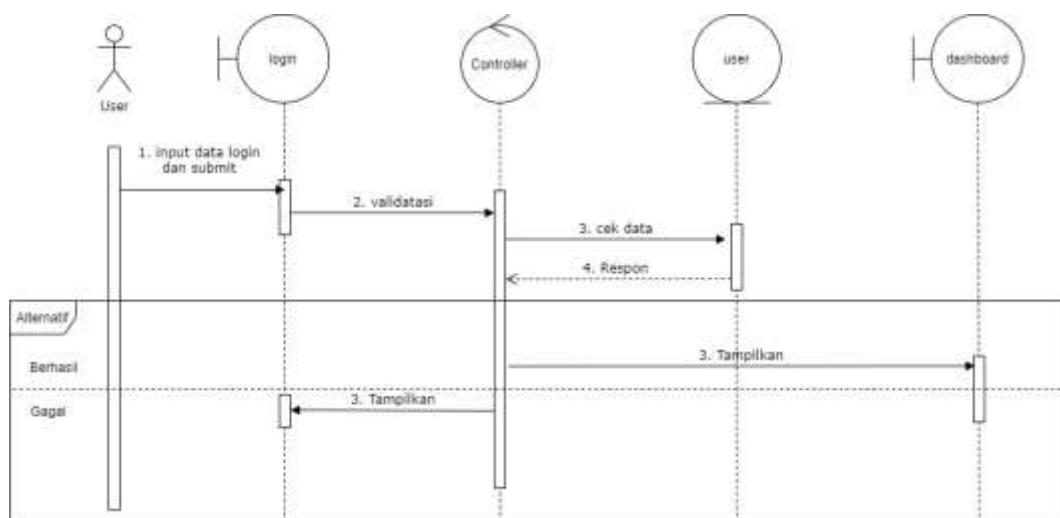
Sequence diagram ini menampilkan proses interaksi antara view, control, entity class saat dilakukannya penginputan data baru user pada sistem. Proses ini dimulai ketika user memilih menu data user pada tampilan dashboard. Setelah itu controller akan mengarahkan user ke tampilan menu user. Setelah berada pada view menu user, user akan menekan tombol tambah data user dan view akan menampilkan form user yang mana pada view ini user bisa menginputkan data. Pada tampilan form ini, user akan mengisi form dengan data user baru yang akan ditambahkan. Setelah itu controller request untuk dilakukannya penambahan data baru pada database. Setelah selesai maka data yang baru ditambahkan akan muncul pada tampilan menu user.



Gambar 4.8 Sequence Diagram input data user

#### 4.1.5.2 Sequence Diagram Login

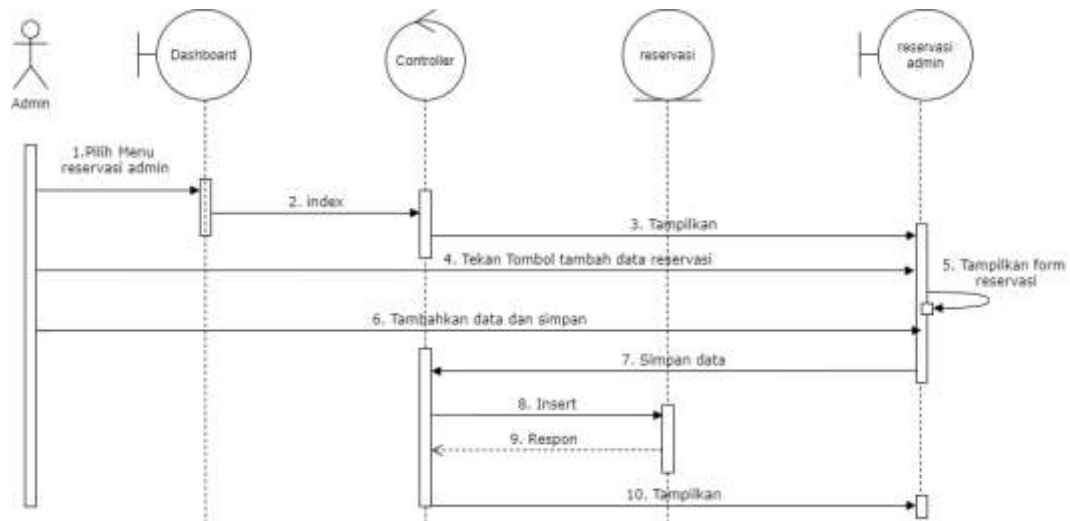
Sequence diagram ini menampilkan proses antara view, control, entity class saat dilakukannya proses login pada system ini. Proses ini dimulai dengan user menginputkan email dan passwordnya pada form login. Setelah selesai, user menekan tombol login dan hasil dari pengisian tersebut akan dilakukan validasi. Sistem akan memberikan respon, jika berhasil maka sistem akan mengarahkan user pada tampilan dashboard dan jika gagal maka sistem akan menampilkan gagal dari pengisian data login.



Gambar 4.9 Sequence diagram login

#### 4.1.5.3 Sequence Diagram Reservasi Admin

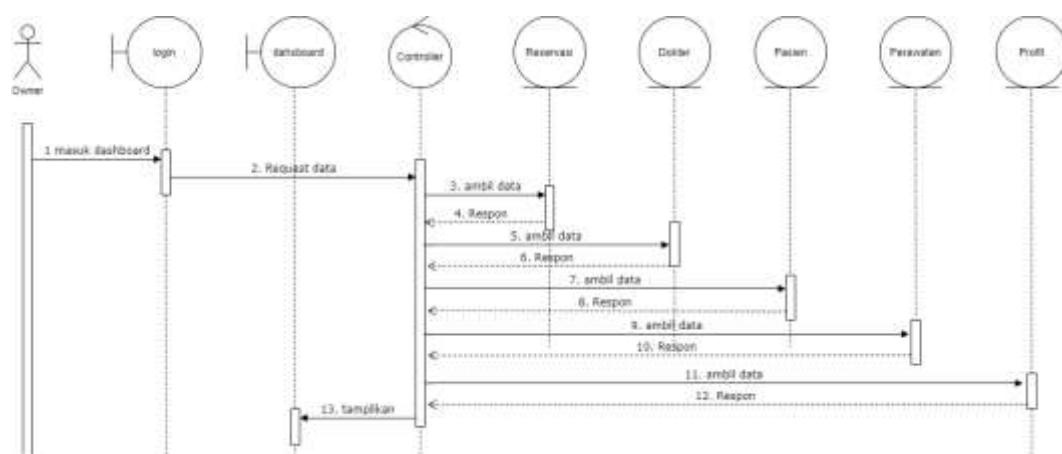
Sequence diagram ini menampilkan proses antara view, control, entity class saat dilakukannya proses reservasi pasien oleh user admin. Proses dimulai dengan admin memilih menu reservasi admin pada tampilan dashboard. Setelah itu controller akan mengarahkan admin ke tampilan reservasi admin. Pada tampilan admin user akan menekan tombol tambah data reservasi dan sistem akan menampilkan form reservasi untuk user menginputkan reservasi baru. Admin mengisikan semua data yang diperlukan pada form dan setelah itu user menekan tombol simpan. Controller akan melakukan request untuk dilakukannya penambahan data baru pada database. Terakhir sistem akan menampilkan list data reservasi yang mana telah ditambahkan data baru.



Gambar 4.10 Sequence diagram reservasi admin

#### 4.1.5.4 Sequence Diagram Menampilkan Dashboard Owner

Sequence diagram ini menampilkan proses antara view, control, entity class saat dilakukannya proses menampilkan dashboard owner. Proses awalnya adalah owner pergi ke halaman login dan melakukan login. Setelah login maka controller akan melakukan request untuk data data yang diperlukan karena pada dashboard terdapat berbagai analisis data. Data yang diambil berupa data reservasi, dokter, pasien, perawatan, dan profit. Setelah data di request maka owner akan diarahkan ke dashboard dan data yang telah diambil akan divisualisasikan pada tampilan dashboard.



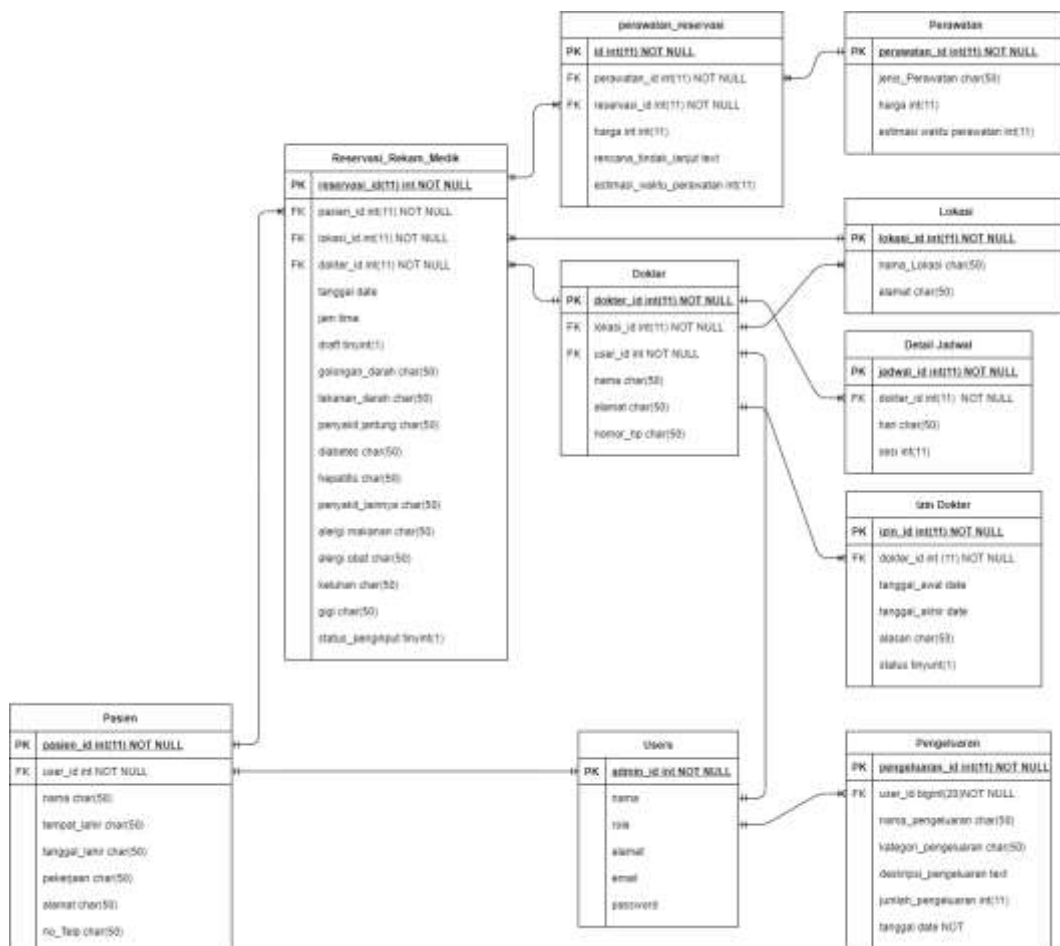
Gambar 4.11 Sequence diagram lihat dashboard owner

## 4.2 Perancangan sistem

Dari tahapan analisis proses yang sedang berjalan, serta alur sistem dan kebutuhan fungsional sistem yang akan dibangun, diperoleh hasil yang menjadi landasan dan tolok ukur untuk merancang sistem. Proses perancangan sistem ini mencakup rancangan basis data, arsitektur aplikasi, class diagram dan perancangan antarmuka.

### 4.2.1 Perancangan Basis data

Struktur tabel menggambarkan setiap tabel yang telah dirancang dalam ERD (Entity Relationship Diagram) dengan menjelaskan semua atribut yang ada di masing-masing tabel. Atribut yang akan diuraikan untuk setiap tabel mencakup nama tabel, primary key, foreign key, nama atribut, tipe data. Berikut ini adalah rincian struktur tabel yang terdapat dalam ERD tersebut.



Gambar 4.12 ERD Sistem Informasi Manajemen Klinik



#### 5.2.2.4 Struktur Tabel Perawatan

Tabel perawatan adalah tabel yang berisikan detail perawatan yang ada pada klinik gigi Xenon Dental House. Pada tabel ini terdapat primary key dengan nama perawatan\_id. Struktur tabel ini dapat dilihat pada tabel 4.1 berikut

Tabel 4.1 Tabel Perawatan

Nama Atribut	Tipe Data	Ukuran	Keterangan
Perawatan_id	int	11	Primary Key
Jenis_perawatan	varchar	50	
Harga	int	11	
estimasi	int	11	

#### 4.2.1.2 Struktur Tabel Lokasi

Tabel lokasi adalah tabel yang berisikan data detail terkait lokasi klinik beserta cabang yang ada pada klinik gigi Xenon Dental House. Tabel ini memiliki primary key dengan nama atribut lokasi\_id. Struktur tabel ini dapat dilihat pada tabel 4.2 berikut.

Tabel 4.2 Tabel Lokasi

Nama Atribut	Tipe Data	Ukuran	Keterangan
Lokasi_id	int	11	Primary Key
Nama_lokasi	varchar	50	
Alamat	char	50	

#### 4.2.1.3 Struktur Tabel Detail jadwal

Tabel detail jadwal adalah tabel yang berisikan detail jadwal dokter yang ada pada klinik gigi Xenon Dental House. Tabel ini memiliki primary key dengan nama atribut jadwal\_id dan foreign key dokter\_id. Struktur tabel ini dapat dilihat pada tabel 4.3 berikut.

Tabel 4.3 Tabel Detail jadwal

Nama Atribut	Tipe Data	Ukuran	Keterangan
Jadwal_id	int	11	Primary Key
Dokter_id	int	11	Foreign Key
hari	varchar	20	
ses	int	11	

#### 4.2.1.4 Struktur Tabel Pengeluaran

Tabel pengeluaran adalah tabel yang berisikan data detail terkait pengeluaran klinik. Tabel ini memiliki primary key dengan nama atribut pengeluaran\_id dan foreign key dengan nama atribut user\_id. Struktur tabel ini dapat dilihat pada tabel 4.4 berikut.

Tabel 4.4 Tabel Pengeluaran

Nama Atribut	Tipe Data	Ukuran	Keterangan
Pengeluaran_id	int	11	Primary Key
User_id	bigint	20	Foreign Key
Nama_pengeluaran	varchar	20	
Kategori_pengeluaran	varchar	20	
Deskripsi_pengeluaran	text		
Jumlah_pengeluaran	int	11	
Tanggal	date		

#### 4.2.1.5 Struktur Tabel Izin Dokter

Tabel izin dokter adalah data yang berisikan data terkait izin dokter dalam bekerja. Tabel izin ini memiliki primary key dengan nama atribut izin\_id dan foreign key user\_id dan lokasi\_id. Struktur tabel 4.5 ini dapat dilihat pada tabel berikut.

Tabel 4.5 Tabel Izin Dokter

Nama Atribut	Tipe Data	Ukuran	Keterangan
Izin_id	int	11	Primary key
Dokter_id	int	11	Foreign Key
Tanggal_awal	date		
Tanggal_akhir	date		
Alasan	varchar	50	
Status	tinyint	1	

#### 4.2.1.6 Struktur Tabel Users

Tabel users adalah tabel yang berisikan detail data detail user. Pada tabel user terdapat primary key dengan nama atribut id. Struktur tabel 4.6 ini dapat dilihat pada tabel berikut.

Tabel 4.6 Tabel Users

Nama Atribut	Tipe Data	Ukuran	Keterangan
id	bigint	20	Primary Key
Nama	varchar	50	
Role	varchar	20	
Email	varchar	255	
Password	varchar	255	
Created_at	Timestamp		
Updated_at	Timestamp		

#### 4.2.1.7 Struktur Tabel Dokter

Tabel dokter adalah tabel yang berisikan data detail dokter. Pada tabel ini terdapat primary key dengan nama atribut dokter\_id dan foreign key dengan nama lokasi\_id. Struktur tabel ini dapat dilihat pada tabel 4.7 berikut.

Tabel 4.7 Tabel Dokter

Nama Atribut	Tipe Data	Ukuran	Keterangan
Dokter_id	int	11	Primary Key
Lokasi_id	int	11	Foreign Key
Nama	varchar	50	
Alamat	varchar	50	
Nomor_hp	varchar	20	

#### 4.2.1.8 Struktur Tabel Perawatan\_reservasi

Tabel perawatan\_reservasi adalah tabel yang menghubungkan antara tabel perawatan dan tabel reservasi. Pada tabel ini terdapat primary key dengan nama atribut id dan foreign key dengan nama harga dan estimasi waktu perawatan. Struktur dari tabel ini dapat dilihat pada tabel 4.8 berikut.

Tabel 4.8 Tabel Perawatan\_reservasi

Nama Atribut	Tipe Data	Ukuran	Keterangan
id	int	11	Primary Key
Perawatan_id	int	11	Foreign Key
Reservasi_id	Int	11	Foreign Key
Harga	int	11	
Estimasi_waktu_perawatan	int	11	
Rencana_tindak_lanjut	text		

#### 4.2.1.9 Struktur Tabel Reservasi\_rekam\_medik

Tabel reservasi\_rekam\_medik adalah tabel yang berisikan reservasi dan rekam medik pasien. Pada tabel ini terdapat primary key dengan nama reservasi\_id dan foreign key dengan nama pasien\_id, lokasi\_id, dan dokter\_id. Struktur tabel ini dapat dilihat pada tabel 4.9 berikut.

Tabel 4.9 Tabel Reservasi Rekam Medik

Nama Atribut	Tipe Data	Ukuran	Keterangan
Reservasi_id	int	11	Primary Key
Pasien_id	int	11	Foreign Key
Lokasi_id	int	11	Foreign Key
Dokter_id	int	11	Foreign Key
Tanggal	date		
Jam	time		
draft	tinyint	1	
Golongan_darah	varchar	20	
Tekanan_darah	varchar	20	
Penyakit_jantung	varchar	20	
Diabetes	varchar	20	
Hepatitis	varchar	20	
Penyakit_lainnya	varchar	20	
Alergi_obat	varchar	20	
Alergi_makanan	varchar	20	
Keluhan	varchar	100	
Gigi	varchar	20	
Status_penginput	tinyint	1	

#### 4.2.1.10 Struktur Tabel Pasien

Tabel pasien adalah tabel yang berisikan data detail terkait pasien. Pada tabel ini terdapat primary key dengan nama pasien\_id dan foreign key dengan nama user\_id. Struktur tabel ini dapat dilihat pada tabel 4.10 berikut.

Tabel 4.10 Tabel Pasien

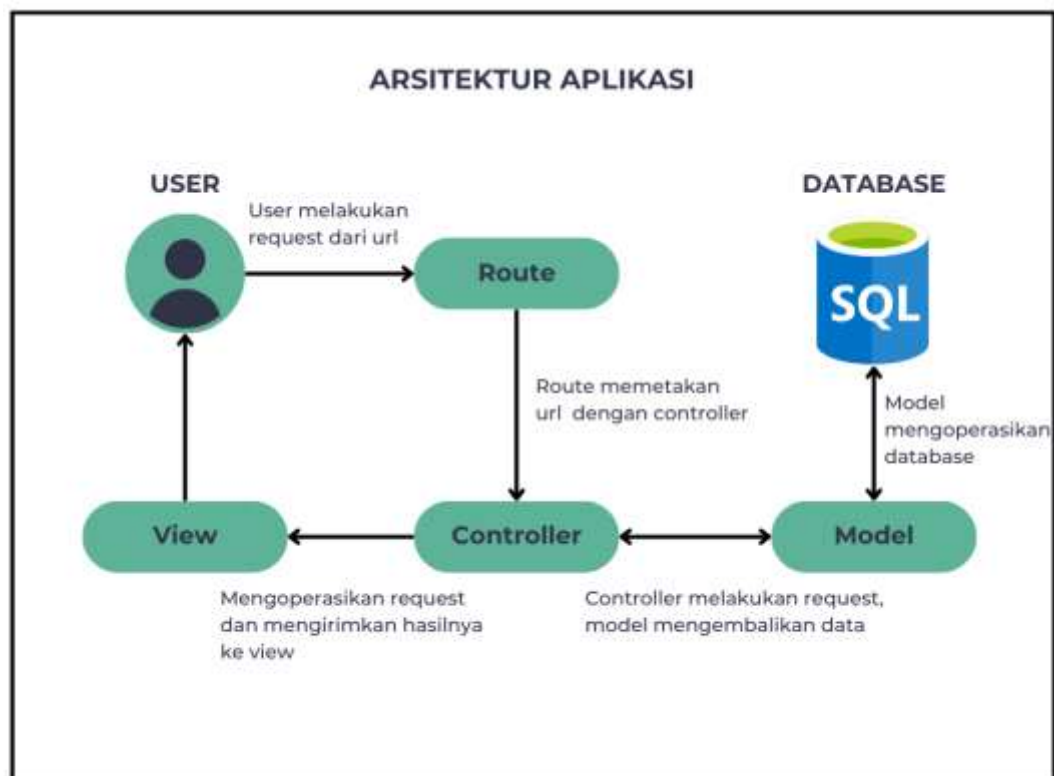
Nama Atribut	Tipe Data	Ukuran	Keterangan
Pasien_id	Int	11	Primary Key
User_id	bigint	20	Foreign Key

Tabel 4.10 Tabel Pasien(lanjutan)

Nama	varchar	50	
Tempat_lahir	varchar	50	
Tanggal_lahir	varchar	20	
Pekerjaan	varchar	50	
Alamat	varchar	20	
No_telp	varchar	20	

#### 4.2.2 Arsitektur Aplikasi

Arsitektur aplikasi yang digunakan dalam penelitian ini mengikuti pola Model-View-Controller (MVC). Pola ini dipilih karena memisahkan logika aplikasi, tampilan, dan manipulasi data, sehingga mempermudah pengembangan dan pemeliharaan aplikasi. Aplikasi ini dikembangkan menggunakan framework Laravel 11 dan menggunakan MySQL sebagai basis data. Gambar pada arsitektur ini bisa dilihat pada gambar 4.13 berikut.



Gambar 4.13 Arsitektur Aplikasi Website

Berikut adalah penjelasan dari setiap komponen dalam arsitektur MVC ini:

1. User (Pengguna):

Pengguna melakukan request dari URL. Ini adalah titik awal di mana pengguna berinteraksi dengan aplikasi web. Setiap permintaan dari pengguna akan diproses oleh sistem melalui serangkaian yang terstruktur.

2. Route:

Route adalah komponen yang bertugas memetakan URL permintaan pengguna ke controller yang sesuai. Dalam Laravel, routing ini didefinisikan dalam file `web.php` atau `api.php`, tergantung pada jenis aplikasi yang dikembangkan. Route memastikan bahwa setiap permintaan dari pengguna diteruskan ke controller yang tepat untuk diproses lebih lanjut.

3. Controller:

Controller bertanggung jawab untuk menerima permintaan dari Route, memprosesnya, dan menentukan apa yang harus dilakukan selanjutnya. Controller melakukan request ke Model untuk mengambil atau memanipulasi data dari database. Setelah mendapatkan data yang diperlukan, Controller akan meneruskannya ke View untuk ditampilkan kepada pengguna.

4. Model:

Model adalah komponen yang berinteraksi langsung dengan database. Model bertugas untuk mengoperasikan database, termasuk mengambil, menyimpan, memperbarui, dan menghapus data.

5. View:

View adalah komponen yang bertugas untuk menampilkan data kepada pengguna. View menerima data dari Controller dan menyajikannya dalam bentuk yang dapat dipahami oleh pengguna. Dalam Laravel, Viewnya menggunakan Blade template engine untuk menampilkan halaman HTML.

## Alur Kerja

1. Pengguna mengirimkan request melalui URL.
2. Route memetakan URL tersebut ke Controller yang sesuai.
3. Controller menerima permintaan dari Route dan meminta data dari Model.
4. Model berinteraksi dengan database MySQL untuk mengambil atau memanipulasi data.
5. Model mengembalikan data ke Controller.
6. Controller mengoperasikan data dan mengirimkannya ke View.
7. View menampilkan data kepada pengguna.

### 4.2.3 Class Diagram

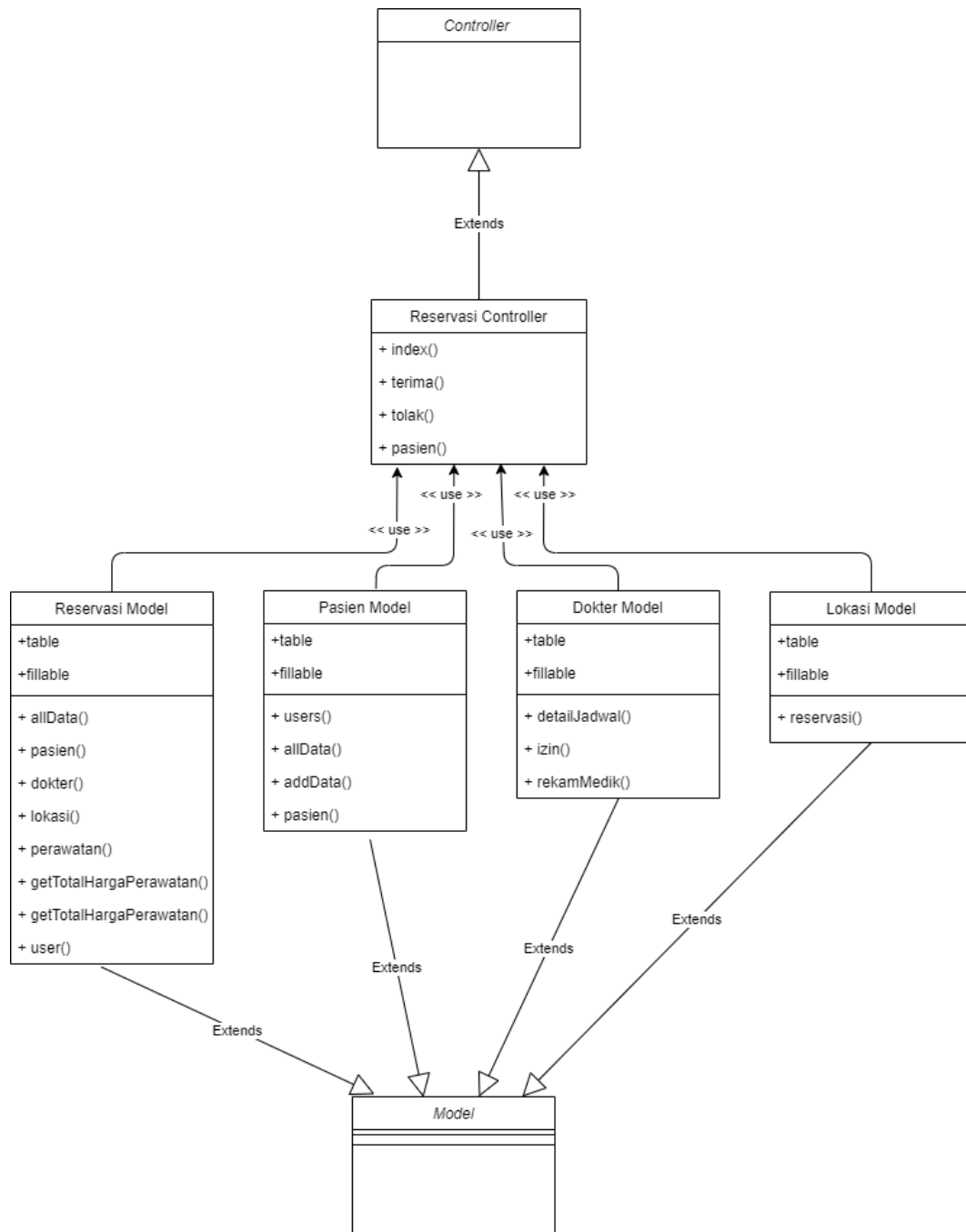
*Class diagram* adalah salah satu jenis diagram dalam *Unified Modeling Language* (UML) yang digunakan untuk menggambarkan struktur dan hubungan antar kelas dalam suatu sistem. Class diagram membantu agar pembangunan sistem yang menggunakan paradigma *Object Oriented Programming* (OOP) menjadi lebih terstruktur, sehingga memudahkan proses pengembangan. Pada pembangunan *website* yang menggunakan *framework* Laravel maka akan terdiri dari kelas Controller dan kelas Model. Struktur kelas Controller terdiri dari fungsi-fungsi yang menjalankan fungsi tertentu, sementara struktur kelas Model terdiri dari atribut dan fungsi agar sistem bisa berinteraksi dengan database. Pada subbab ini akan dijelaskan class diagram yang terdapat pada sistem ini meliputi *class diagram* admin konfirmasi reservasi dari pasien, *class diagram* pengajuan permintaan izin dokter, serta *class diagram* pemilik melihat jumlah kunjungan pasien.

#### 5.2.2.4 Class Diagram Konfirmasi Reservasi Dari Pasien

*Class diagram* admin konfirmasi reservasi dari pasien adalah *class diagram* yang digunakan dalam proses admin menerima atau menolak reservasi yang diajukan dari calon pasien. Pada class diagram ini terdapat 1 kelas reservasi *controller* yang memiliki fungsi *index()*, *terima()*, *tolak()*, *pasien()*. Pada diagram



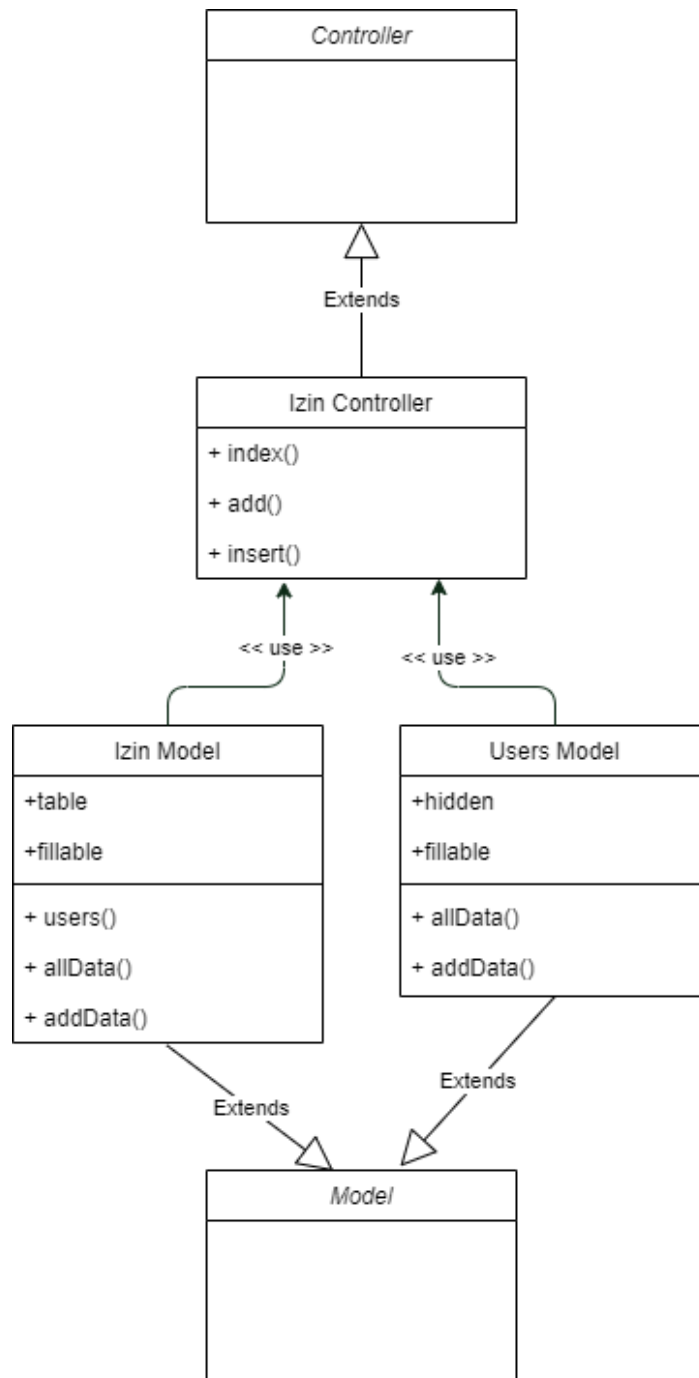
ini juga terdaoat 4 kelas model diantaranya reservasi model, pasien model, dokter model, lokasi model. *Class diagram* ini dapat dilihat pada gambar 4.14



Gambar 4.14 Class Diagram Konfirmasi Reservasi Dari Pasien

#### 4.2.3.2 Class Diagram Pengajuan Permintaan Izin Dokter

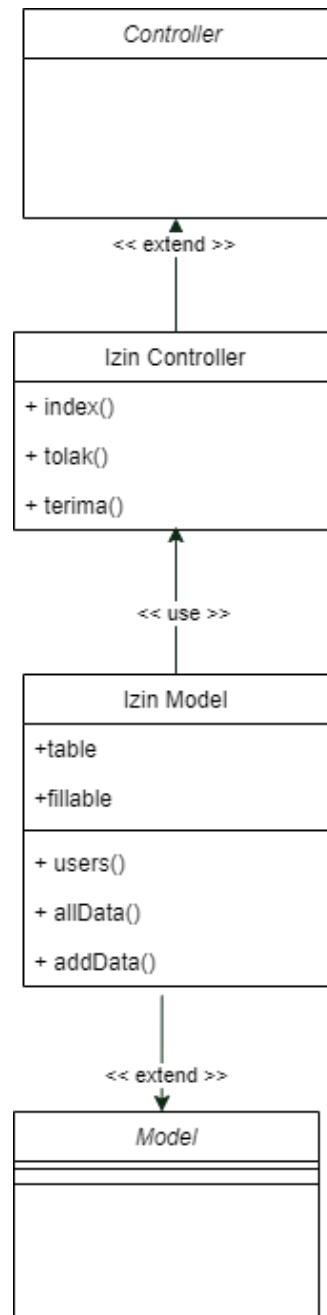
Class diagram pengajuan permintaan izin dokter adalah *class diagram* yang digunakan dalam proses permintaan izin dokter kepada *owner*. Pada diagram ini terdapat kelas izin dokter yang memiliki fungsi `index()`, `add()` dan `insert()`. *Class diagram* ini bisa dilihat pada gambar 4.15



Gambar 4.15 Class Diagram Pengajuan Permintaan Izin Dokter

#### 4.2.3.3 Class Diagram Pemilik Memutuskan Perizinan Dokter

Class diagram pemilik memutuskan perizinan dokter adalah *class diagram* yang digunakan dalam proses menolak atau menerima perizinan yang diajukan dokter kepada owner. Diagram ini memiliki kelas izin yang memiliki fungsi `index()`, `tolak()`, dan `terima()`. Diagram ini juga memiliki 1 kelas model yaitu `izin model`. Class diagram ini bisa dilihat pada gambar 4.16 berikut.



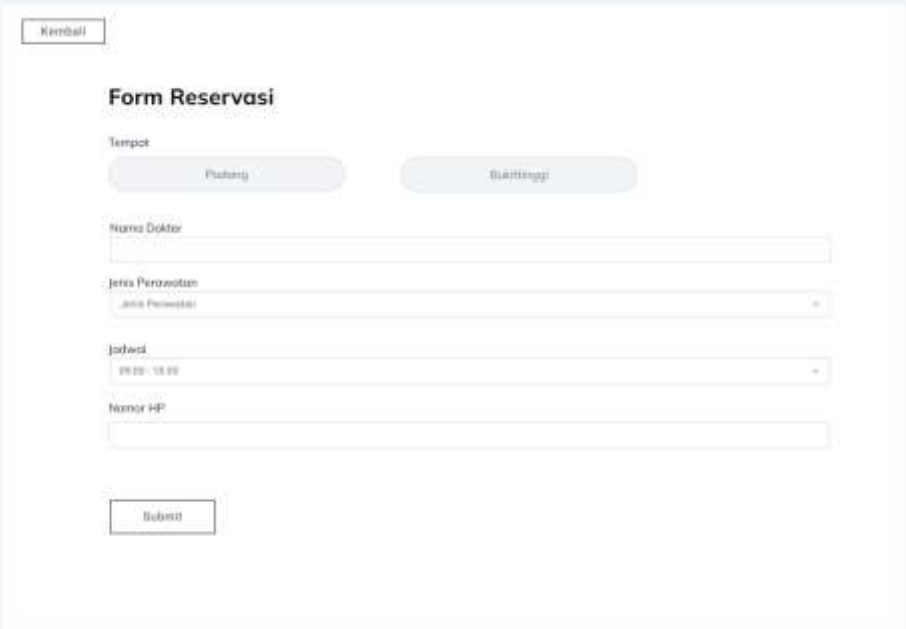
Gambar 4.16 Class Diagram Pemilik Memutuskan Perizinan Dokter

#### 4.2.4 Tampilan Mockup Antarmuka

Mockup antarmuka merupakan bentuk visual desain antarmuka pengguna yang memberikan gambaran detail mengenai tata letak elemen-elemen yang ada di dalam suatu *website*. Dalam sub bab ini akan dijelaskan desain antarmuka dari form reservasi, tampilan data user, dashboard owner, dan landing page *website*.

##### 4.2.4.1 Mockup Antarmuka Halaman Form Reservasi

Mockup antarmuka halaman form reservasi adalah halaman yang berfungsi untuk menerima inputan berupa informasi informasi yang dibutuhkan dalam melakukan reservasi. Form reservasi ini bisa diisi melalui akun admin dan pasien. Pada halaman ini user akan menginputkan tempat reservasi, dokter, jenis perawatan, jadwal perawatannya. Tampilan halaman ini dapat dilihat pada gambar 4.17 berikut.

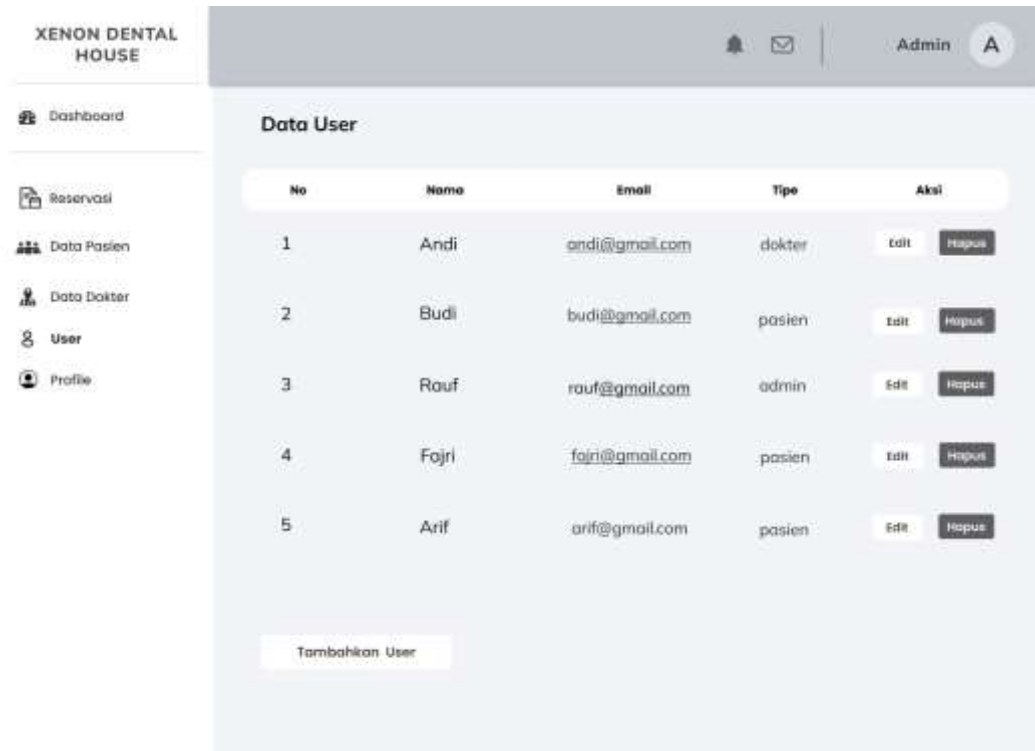


The image shows a web form titled "Form Reservasi" with a light blue header bar. At the top left of the form is a "Kembali" (Back) button. Below the title, there are two buttons: "Pasien" and "Buattinggi". The form contains several input fields: "Nama Dokter" (Doctor Name), "Jenis Perawatan" (Type of Treatment), "Jadwal" (Schedule), and "Nomor HP" (Phone Number). Each input field has a small "x" icon on the right side. At the bottom of the form is a "Simpan" (Save) button.

Gambar 4.17 Mockup Form Reservasi

#### 4.2.4.2 Mockup Antarmuka Halaman Data User

Mockup antarmuka halaman data user adalah halaman yang mana data user dikelola. Pada halaman ini terdapat list user *website* ini yang dapat dikelola oleh admin. Pada halaman ini admin dapat menghapus data, menambahkan, serta dapat mengedit data user yang ada pada klinik ini. Tampilan ini dapat dilihat pada gambar yang tertera pada gambar 4.18 berikut.

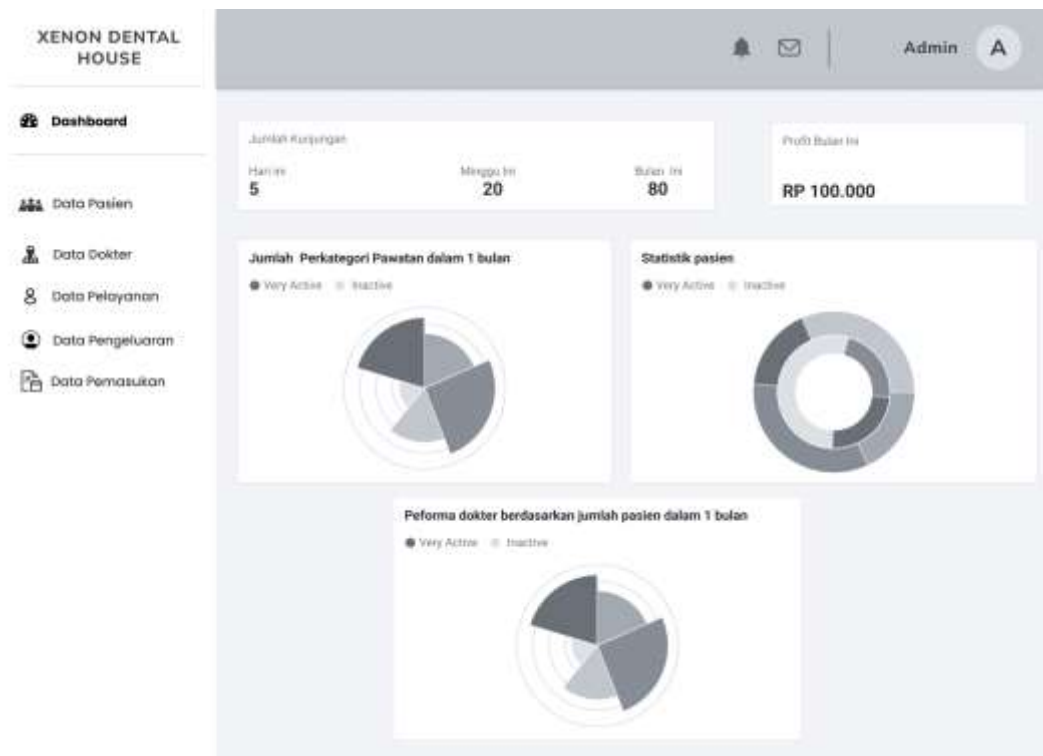


Gambar 4.18 Mockup Tampilan data user

#### 4.2.4.3 Mockup Antarmuka Halaman Dashboard owner

Mockup antarmuka halaman dashboard adalah halaman yang menampilkan visualisasi data terhadap owner klinik Xenon Dental House. Data yang ditampilkan ini adalah data hasil analisis berupa jumlah pasien yang diterima dokter selama 1 bulan, pasien berdasarkan umur dari pasien, perawatan yang dilakukan pada pasien, serta jumlah kunjungan pasien. Pada halaman ini owner juga dapat melakukan pemilihan pada visualiasi untuk menampilkan bulan apa yang ingin

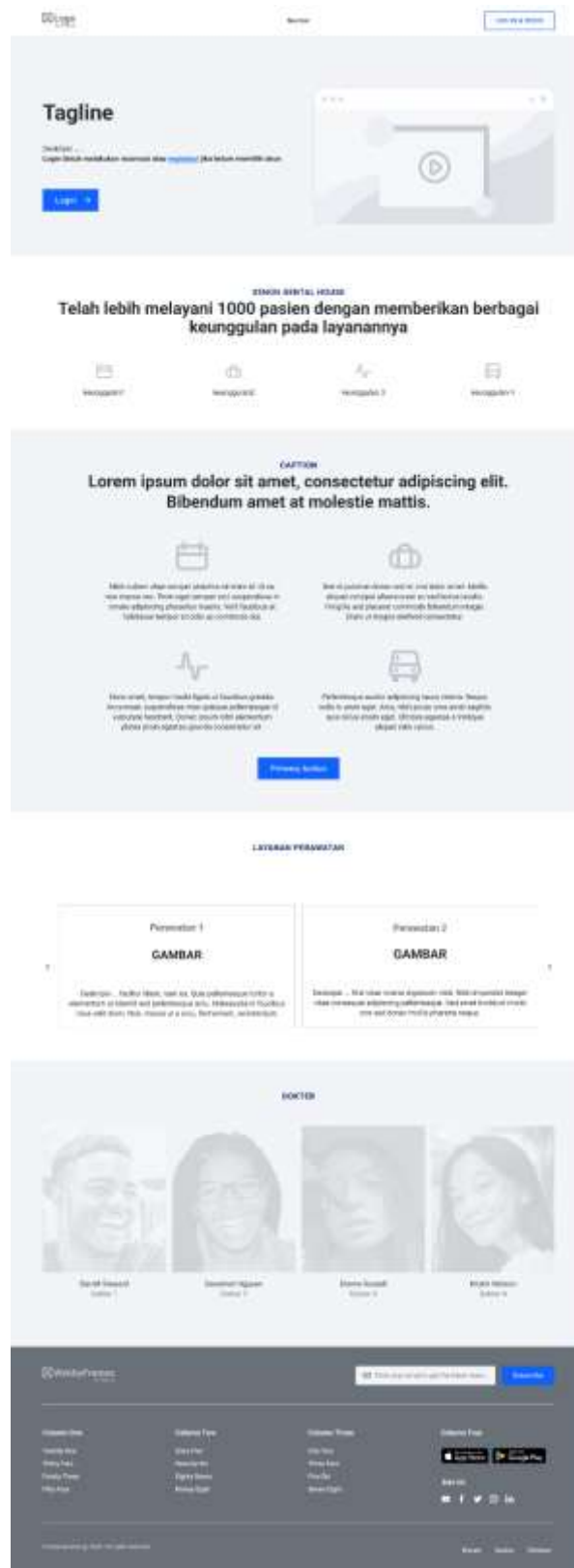
ditampilkan, sehingga owner dapat memfilter bulan apa yang ingin ditampilkan. Tampilan ini dapat dilihat pada gambar 4.19 berikut



Gambar 4.19 Mockup Dashboard akun owner

#### 4.2.4.4 Mockup Antarmuka Halaman Landing page

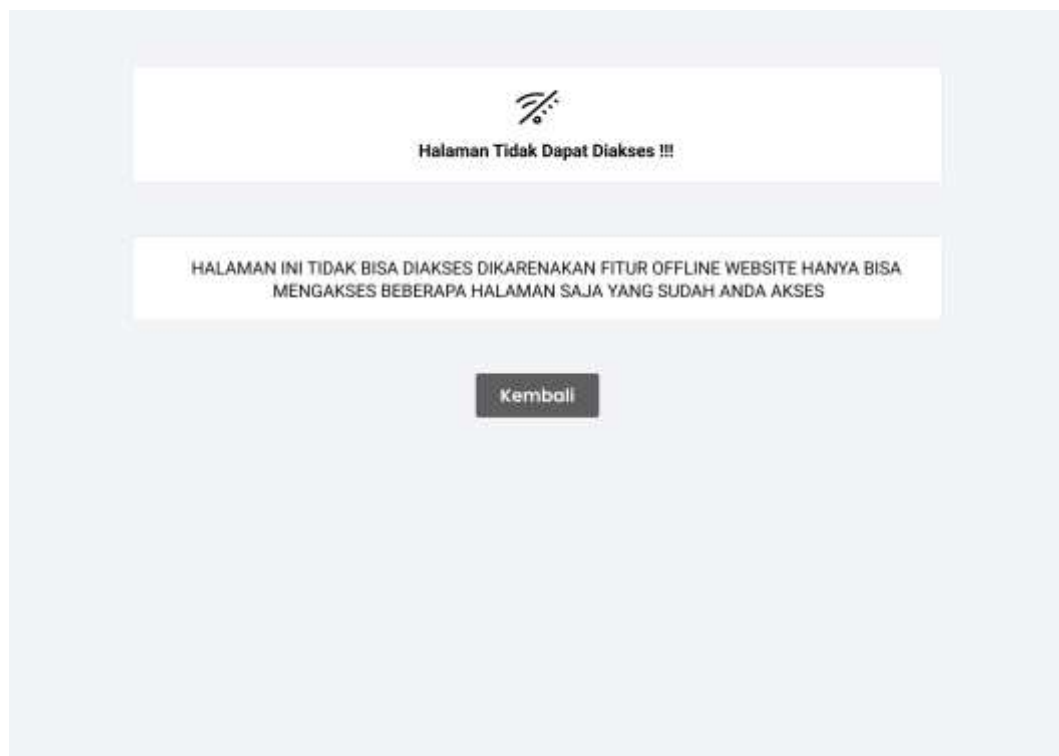
Mockup antarmuka halaman landing page adalah halaman yang menampilkan landing page dari *website* klinik gigi Xenon Dental House. Landing page adalah halaman tunggal yang dirancang untuk pemasaran dan branding dari *website* sehingga diharapkan dapat menarik pengunjung untuk keuntungan dari klinik seperti pada klinik ini yaitu melakukan reservasi untuk perawatan di klinik ini. Hasil dari mockup ini dapat dilihat pada gambar 4.20 berikut.



Gambar 4.20 Mockup Landing page *website*

#### 4.2.4.5 Mockup Antarmuka Offline PWA

Mockup antarmuka offline PWA adalah halaman yang menampilkan tampilan ketika user mencoba mengakses website yang belum pernah diaksesnya setelah server mati. Salah satu keuntungan dari implementasi PWA adalah user bisa mengakses tampilan website meskipun server mati atau koneksi ke server terputus. Fitur *offline* ini berlaku ketika user sudah pernah mengakses url yang akan diakses ini sebelumnya, jika belum pernah maka tampilan ini yang akan ditampilkan oleh website. Mockup ini dapat dilihat pada gambar 4.21 berikut.



Gambar 4.20 Mockup Antarmuka Offline PWA



## **BAB V**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **6.1 Implementasi Sistem**

Sistem informasi manajemen klinik Xenon Dental House dengan menggunakan teknologi *website* ini difungsikan sebagai sistem yang dapat mengelola dan menampilkan data dan informasi yang berkaitan dengan manajemen klinik dimulai dari reservasi, izin dokter, pengelolaan data layanan perawatan, hingga analisis data klinik. Pada sistem ini terdapat 4 aktor yang terlibat dalam 29 fungsional yang diharapkan dapat membantu terhadap semua proses pada sistem.

Implementasi *website* ini dilakukan dengan menggunakan perangkat keras dengan spesifikasi berikut:

1. Komputer dengan processor Intel Core i5-1035G1
2. Random Access Memory (RAM) 8GB
3. Penyimpanan dengan kapasitas 476.94 GB

Spesifikasi perangkat lunak yang digunakan dalam implementasi sistem ini adalah sebagai berikut:

1. Sistem operasi windows 10
2. Web browser yang digunakan adalah google chrome versi
3. Code editor visual studio code
4. Web server Built-in PHP Development Server
5. Laravel 11 dengan php 8.2
6. Database MySQL

##### **6.1.1 Pengkodean sistem**

Bagian ini menjelaskan tentang implementasi desain aplikasi dan arsitektur aplikasi yang telah dibahas pada bab sebelumnya dalam bentuk kode program. Beberapa potongan kode program yang akan diuraikan dalam bagian ini meliputi kode bagian route, controller, model dan view pada sistem ini.

### 6.1.1.1 Kode Program Route

Routing adalah proses yang mengarahkan permintaan HTTP ke bagian tertentu dari aplikasi. Dalam Laravel, routing berfungsi untuk memetakan URL yang diminta pengguna ke kontroler tertentu dalam aplikasi. Hal ini memungkinkan aplikasi web untuk merespons permintaan pengguna dengan cara yang sesuai, seperti menampilkan halaman, memproses request, atau mengelola data. Pada routing ini terdapat beberapa metode yang sering digunakan yaitu seperti metode get, post, delete. Pada dasarnya terdapat beberapa jenis route pada Laravel sebagai berikut.

1. Route dasar yang merupakan pemetaan url ke fungsi penganan
2. Route dengan controller yang merupakan pemetaan url ke controller
3. Route group yang merupakan pemetaan kelompok yang memiliki kebutuhan middleware yang sama
4. Route nama yang merupakan route yang diberikan nama untuk referensi yang lebih mudah

Pada bagian ini akan dijelaskan 2 potongan kode route yang memiliki middleware yang berbeda yaitu role untuk user dokter dan route user admin yang dapat dilihat pada gambar 5.1 dan 5.2 berikut .

```
// Routes untuk dokter

Route::middleware(['role:Dokter'])->group(function () {
    Route::get('/dokter', [DokterHomeController::class, 'index'])->name('dokter.index');
    Route::get('/dataabsen', [IzinController::class, 'index'])->name('dataabsen.index');
    Route::get('/dataabsen/add', [IzinController::class, 'add'])->name('dataabsen.add');
    Route::get('/dataabsen/edit/{id}', [IzinController::class, 'edit'])->name('dataabsen.edit');
    Route::post('/dataabsen/update/{id}', [IzinController::class, 'update'])->name('dataabsen.update');
    Route::post('/dataabsen/insert', [IzinController::class, 'insert'])->name('dataabsen.insert');
    Route::delete('/dataabsen/destroy/{id}', [IzinController::class, 'destroy'])->name('dataabsen.destroy');

    Route::get('/ddatapasien', [PasienController::class, 'index'])->name('ddatapasien.index');
    Route::get('/ddatapasien/add', [PasienController::class, 'add'])->name('ddatapasien.add');
    Route::post('/ddatapasien/insert', [PasienController::class, 'insert'])->name('ddatapasien.insert');
    Route::get('/ddatapasien/edit/{id}', [PasienController::class, 'edit'])->name('ddatapasien.edit');
    Route::post('/ddatapasien/update/{id}', [PasienController::class, 'update'])->name('ddatapasien.update');
    Route::delete('/ddatapasien/destroy/{id}', [PasienController::class, 'destroy'])->name('ddatapasien.destroy');
    Route::get('/ddatapasien/rekam/{id}', [RekamController::class, 'index'])->name('dokter.rekamkasien');
    Route::get('/ddatapasien/rekam/edit/{id}', [RekamController::class, 'edit'])->name('dokter.rekamkasien.edit');
    Route::post('/ddatapasien/rekam/update/{id}', [RekamController::class, 'update'])->name('dokter.rekamkasien.update');
});
```

Gambar 5.1 Potongan Kode Route Dokter

```

> web.php > Closure
Router::middleware(['role:admin'])->group(function () {
    // Routes untuk DataDokter
    Route::get('/datadokter', [DataDokterController::class, 'index'])->name('datadokter.index');
    Route::get('/datadokter/add', [DataDokterController::class, 'add'])->name('datadokter.add');
    Route::post('/datadokter/insert', [DataDokterController::class, 'insert'])->name('datadokter.insert');
    Route::get('/datadokter/edit/{id}', [DataDokterController::class, 'edit'])->name('datadokter.edit');
    Route::post('/datadokter/update/{id}', [DataDokterController::class, 'update'])->name('datadokter.update');
    Route::delete('/datadokter/destroy/{id}', [DataDokterController::class, 'destroy'])->name('datadokter.destroy');

    // Routes untuk Jadwal DataDokter
    Route::get('/datadokter/jadwal/{id}', [JadwalController::class, 'index'])->name('dokterjadwal.index');
    Route::get('/datadokter/jadwal/add/{id}', [JadwalController::class, 'add'])->name('dokterjadwal.add');
    Route::post('/datadokter/jadwal/insert/{id}', [JadwalController::class, 'insert'])->name('dokterjadwal.insert');
    Route::get('/datadokter/jadwal/edit/{id}', [JadwalController::class, 'edit'])->name('dokterjadwal.edit');
    Route::post('/datadokter/jadwal/update/{id}', [JadwalController::class, 'update'])->name('dokterjadwal.update');
    Route::delete('/datadokter/jadwal/destroy/{id}', [JadwalController::class, 'destroy'])->name('dokterjadwal.destroy');

    // Routes untuk Reservasi Admin
    Route::get('/reservasi/admin/', [ReservasiController::class, 'index'])->name('reservasi.index');
    Route::get('/reservasi/admin/add/', [ReservasiController::class, 'add'])->name('reservasi.add');
    Route::post('/reservasi/admin/insert/', [ReservasiController::class, 'insert'])->name('reservasi.insert');
    Route::get('/reservasi/admin/edit/{id}', [ReservasiController::class, 'edit'])->name('reservasi.edit');
    Route::post('/reservasi/admin/update/{id}', [ReservasiController::class, 'update'])->name('reservasi.update');
    Route::delete('/reservasi/admin/destroy/{id}', [ReservasiController::class, 'destroy'])->name('reservasi.destroy');
    Route::post('/reservasi/admin/terima/{id}', [ReservasiController::class, 'terima'])->name('reservasi.terima');
    Route::post('/reservasi/admin/tolak/{id}', [ReservasiController::class, 'tolak'])->name('reservasi.tolak');
    Route::get('/reservasi/pasien/', [ReservasiController::class, 'pasien'])->name('reservasi.pasien');
    Route::get('/api/dokter-by-lokasi', [ReservasiController::class, 'getDokterByLokasi']);
    Route::get('/api/unavailable-times', [ReservasiController::class, 'getUnavailableTimes']);
    Route::get('/api/available-dates', [ReservasiController::class, 'getAvailableDates']);
    Route::get('/api/available-days', [ReservasiController::class, 'getAvailableDays']);
    Route::get('/api/booked-times', [ReservasiController::class, 'getBookedTimes']);

    Route::get('/api/sesi-by-dokter-and-hari', [ReservasiController::class, 'getSesiByDokterAndHari']);
    Route::get('/reservasi/getDoctors', [ReservasiController::class, 'getDokterByLokasi'])->name('reservasi.getDoctors');
    Route::get('/reservasi/getTimeSlots', [ReservasiController::class, 'getTimeSlots'])->name('reservasi.getTimeSlots');
});

```

Gambar 5.2 Potongan Kode Route admin

### 6.1.1.2 Kode Program Controller

Controller adalah bagian yang bertanggung jawab untuk mengelola logika aplikasi dan merespons permintaan HTTP yang masuk. Controller menghubungkan rute (routing) dengan tampilan (views) serta berinteraksi dengan model untuk mengambil dan manipulasi data, dan bertindak sebagai pengatur alur aplikasi. Controller pada laravel pada umumnya memiliki metode standar seperti index() untuk menampilkan tampilan awal, create() untuk tampilan tempat penambahan data, store() untuk menambahkan data, show() untuk menampilkan detail dari tampilan index, edit() untuk tampilan tempat pengeditan data serta update() untuk memperbarui data yang sudah ada. Pada bagian ini akan dijelaskan controller untuk datadoktercontroller dan jadwalcontroller.

Datadoktercontroller adalah controller yang memproses data pada tabel dokter seperti data nama, alamat, nomor\_hp, dan lokasi\_id. Controller ini dapat menambahkan, menghapus, memperbarui serta dapat menghapus data dari tabel dokter. Sedangkan untuk jadwalcontroller adalah controller yang 57iway mengatur

proses yang dilakukan pada jadwal dokter. Controller ini juga dapat melakukan CRUD pada tabel detail jadwal yang memiliki atribut hari, sesi dan dokter id. Kode program ini dapat dilihat pada gambar 5.3 dan gambar 5.4 berikut.

```
class DataDokterController extends Controller

    public function add()
    {
        $lokasi = Lokasi::all();
        return view('admin.datadokter_add', compact('lokasi'));
    }

    public function insert(Request $request)
    {
        $request->validate([
            'nama' => 'required',
            'alamat' => 'required',
            'nomor_hp' => 'required',
            'lokasi_id' => 'required',
        ], [
            'nama.required' => 'Nama harus diisi!',
            'alamat.required' => 'alamat harus diisi!',
            'nomor_hp.required' => 'nomor_hp harus diisi!',
            'lokasi_id.required' => 'lokasi praktek harus diisi'
        ]);

        $data = [
            'nama' => $request->nama,
            'alamat' => $request->alamat,
            'nomor_hp' => $request->nomor_hp,
            'lokasi_id' => $request->lokasi_id
        ];

        $this->datadokter->addData($data);
        Alert::success('Berhasil!', 'Data dokter berhasil ditambahkan!');
        return redirect('/datadokter');
    }

    public function destroy($dokter_id)
    {
        DB::table('dokter')->where('dokter_id', $dokter_id)->delete();

        // Menampilkan alert sukses dan mengarahkan kembali ke halaman data dokter
        Alert::success('Berhasil!', 'Data dokter berhasil dihapus!');
        return redirect('/datadokter');
    }
}
```

Gambar 5.3 Potongan kode DataDokterController

```

class JadwalController extends Controller
{
    public function insert(Request $request, $id)
    {
        // Validasi input form
        $request->validate([
            'hari' => 'required',
            'sesi' => 'required',
        ], [
            'hari.required' => 'Hari harus diisi!',
            'sesi.required' => 'Sesi harus diisi!',
        ]);

        // Menyiapkan data yang akan disimpan
        $data = [
            'hari' => $request->hari,
            'sesi' => $request->sesi,
            'dokter_id' => $id,
        ];

        // Menyimpan data menggunakan model (asumsi ada method addData di model yang digunakan)
        $this->datadokter->addData($data);

        // Menampilkan pesan sukses menggunakan SweetAlert (asumsi Alert diimport dengan benar)
        Alert::success('Berhasil!', 'Data jadwal berhasil ditambahkan!');

        // Redirect ke halaman yang diinginkan dengan sintaks kurung keriting ganda
        return redirect("/datadokter/jadwal/{$id}");
    }

    public function destroy($jadwal_id)
    {
        // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
        DB::table('detail_jadwal')->where('jadwal_id', $jadwal_id)->delete();

        // Menampilkan alert sukses dan mengarahkan kembali ke halaman data dokter
        Alert::success('Berhasil!', 'Data jadwal dokter berhasil dihapus!');
        $id_dokter = DB::table('detail_jadwal')->where('jadwal_id', $jadwal_id)->value('dokter_id');
        if($id_dokter==NULL){
            return redirect("/datadokter");
        }
        else{
            return redirect("/datadokter/jadwal/{$id_dokter}");
        }
    }
}

```

Gambar 5.4 Potongan kode DataDokterController

### 6.1.1.3 Kode Program Model

Model adalah komponen dari arsitektur MVC (Model-View-Controller) yang bertugas untuk berinteraksi dengan basis data. Model merepresentasikan tabel dalam basis data dan menyediakan cara untuk mengakses serta memanipulasi data dalam tabel tersebut. Pada bagian ini akan dijelaskan 2 model yaitu model detail jadwal dan model dokter. Pada model ini terdapat atribut khusus seperti \$table yang merepresentasikan nama tabel dari database, \$primary\_key yang merepresentasikan primary key dari tabel dan \$fillable yang merupakan atribut dari tabel yang bisa diisi dengan data.

Model detailjadwal adalah model yang merepresentasikan tabel jadwal dokter pada database yang memiliki primary key jadwal\_id serta memiliki atribut lain seperti dokter\_id, hari dan sesi. Model ini memiliki hubungan many to one pada tabel dokter yang dapat dilihat pada fungsi dokter(). Model ini juga memiliki fungsi allData yang mana akan melakukan request semua data jadwal ke database dan memiliki fungsi addData yang akan menambahkan data ke tabel detail jadwal pada database. Kode program dari model ini dapat dilihat pada gambar 5.5.

Model dokter adalah model yang merepresentasikan tabel dokter pada database yang memiliki primary key dokter\_id serta memiliki atribut lain seperti nama, alamat, nomor\_hp, lokasi\_id dan user\_id. Model ini memiliki hubungan one to many pada tabel detail jadwal yang dapat dilihat pada fungsi detail jadwal dan memiliki fungsi izin yang memperlihatkan hubungan hasmany kepada tabel izin. Model ini juga memiliki fungsi allData yang mana akan melakukan request semua data dokter ke database dan memiliki fungsi addData yang akan menambahkan data ke tabel dokter. Kode program dari model ini dapat dilihat pada gambar 5.6 dibawah ini.

```

class DetailJadwal extends Model
{
    use HasFactory;

    protected $table = 'detail_jadwal';
    protected $primaryKey = 'jadwal_id';
    public $timestamps = false;

    protected $fillable = [
        'dokter_id',
        'hari',
        'sesi'
    ];

    public function dokter()
    {
        return $this->belongsTo(Dokter::class, 'dokter_id');
    }

    public function allData()
    {
        return DB::table('detail_jadwal')->get();
    }

    public function addData($data)
    {
        DB::table('detail_jadwal')->insert($data);
    }
}

```

Gambar 5.5 Potongan kode model detail jadwal

```

class Dokter extends Model
{
    use HasFactory;

    protected $table = 'dokter';
    protected $primaryKey = 'dokter_id';
    public $timestamps = false;

    protected $fillable = [
        'nama',
        'alamat',
        'nomor_hp',
        'lokasi_id',
        'user_id',
    ];

    public function detailJadwal()
    {
        return $this->hasMany(DetailJadwal::class, 'dokter_id');
    }

    public function izin()
    {
        return $this->hasMany(Izin::class, 'dokter_id');
    }

    public function rekamMedik()
    {
        return $this->hasMany(RekamMedik::class, 'dokter_id');
    }
}

```

Gambar 5.6 Potongan kode model dokter

#### 6.1.1.4 Kode Program View

View adalah komponen dari arsitektur MVC (Model-View-Controller) yang bertanggung jawab untuk menampilkan data kepada pengguna. View adalah tempat di mana menentukan bagaimana data yang dikirim dari controller ke pengguna akhir akan ditampilkan. Pada bagian ini akan dijelaskan 2 view yaitu view `datadokter_add` dan `datadokter` yang menggunakan template engine bawaan Laravel yaitu file blade. View `datadokter_add` adalah view yang menampilkan form saat akan ditambahkan data dokter baru oleh admin. Pada view ini admin akan



diminta mengisi nama, alamat, nomor\_hp dan lokasi\_id dari dokter. View datadokter adalah view yang menampilkan data dokter setelah ditambahkan data dokter oleh admin. View ini menampilkan no,nama, dan alamat dari dokter. View ini juga menyediakan tombol dan link yang akan mengarahkan admin ke tampilan detail jadwal dari dokter, menghapus data dokter, menambahkan serta mengedit data dokter. Kedua view ini dapat dilihat pada gambar 5.7 dan gambar 5.8 berikut.

```
@extends('layout.v_template')

@section('main-content')
<h3 class="font-weight-bold mx-4">Tambah data Dokter </h3>
<!-- Page Heading -->
<div class="mx-4">
    <form action="/datadokter/insert" method="post" enctype="multipart/form-data">
        @csrf
        <div class="form-group">
            <label>Nama Dokter</label>
            <input name="nama" class="form-control @error('nama') is-invalid @enderror" value="{{ old('nama') }}">
            <div class="invalid-feedback">
                @error('nama')
                {{ $message }}
            @enderror
        </div>

        <div class="form-group">
            <label>Alamat</label>
            <input name="alamat" class="form-control @error('alamat') is-invalid @enderror" value="{{ old('alamat') }}">
            <div class="invalid-feedback">
                @error('alamat')
                {{ $message }}
            @enderror
        </div>

        <div class="form-group">
            <label>Nomor Hp </label>
            <input name="nomor_hp" class="form-control @error('nomor_hp') is-invalid @enderror" value="{{ old('nomor_hp') }}">
            <div class="invalid-feedback">
                @error('nomor_hp')
                {{ $message }}
            @enderror
        </div>
    </form>
</div>
```

Gambar 5.7 Potongan kode view datadokter\_add

```

@extends('layout.v_template')

@section('main-content')
<!-- Page Heading -->
<h3 class="font-weight-bold">Data Dokter </h3>

<table id="example" class="table table-striped table-bordered mt-2" style="width:100%">
  <thead>
    <tr>
      <th>no </th>
      <th>Nama</th>
      <th>No HP</th>
      <th>Jadwal</th>
      <th>Aksi</th>
    </tr>
  </thead>
  <tbody>

    @foreach ($datadokter as $dok)
    <tr>
      <td scope="row">{{ $loop->iteration }}</td>
      <td>{{ $dok->nama }}</td>

      <td>
        @php
          $nomor_hp = $dok->nomor_hp;
          if (substr($nomor_hp, 0, 2) == '88') {
            $nomor_hp = '628' . substr($nomor_hp, 2);
          }
        @endphp
        {{ $nomor_hp }}
      </td>

      <td><a href="/datadokter/jadwal/{{ $dok->dokter_id }}">Detail Jadwal</a></td>
      <td>
        <div class="d-flex">
          <a href="/datadokter/edit/{{ $dok->dokter_id }}" class="btn btn-sm btn-primary mr-2">Edit</a>
          <form action="{{ route('datadokter.destroy', [$dok->dokter_id]) }}" method="POST" display="inline"
            onsubmit="return confirm('Yakin ingin menghapus? Data yang berhubungan dengan data dokter ini :
            @csrf
            @method('DELETE')
            <button type="submit" class="btn btn-danger">Hapus</button>
          </form>
        </div>
      </td>
    </tr>
    @endforeach
  </tbody>
</table>

```

Gambar 5.8 Potongan kode View datadokter

#### 6.1.1.5 Kode Program *Service Worker* dan *Manifest* untuk PWA

Dalam penerapan mode *offline PWA* pada laravel diperlukan 2 file khusus yang bernama `service_worker.js` dan `manifest.json`. *Service worker* adalah file *JavaScript* yang berjalan di *background browser*, terpisah dari halaman web. Ini bertindak sebagai perantara aplikasi web dan jaringan yang memungkinkan *website* untuk melakukan *caching* konten, pengelolaan permintaan jaringan dan akses *offline website*. Manifest file adalah file yang memberikan informasi tentang aplikasi agar bisa diinstall seperti aplikasi native dengan ikon, nama, warna tema, dan pengaturan tampilan. Pada manifest terdapat atribut seperti nama aplikasi ketika diinstall, nama singkat, url awalnya, display dan background. Potongan kode dari service worker dan anifest bisa dilihat pada gambar 5.9 dan gambar 5.10 berikut.

```

self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        return cache.addAll(urlsToCache);
      })
  );
});

self.addEventListener('fetch', event => {
  event.respondWith(
    fetch(event.request).then(response => {
      // Jika berhasil mengambil dari jaringan, simpan di cache dan kembalikan respons
      return caches.open(CACHE_NAME).then(cache => {
        cache.put(event.request, response.clone());
        return response;
      });
    }).catch(() => {
      // Jika jaringan gagal, coba ambil dari cache
      return caches.match(event.request).then(response => {
        return response || caches.match('/offline');
      });
    })
  );
});

```

Gambar 5.9 Kode serviceworker.js

```

{
  "name": "Your App Name",
  "short_name": "AppName",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#000000",
  "icons": [
    {
      "src": "/images/icons/icon-72x72.png",
      "sizes": "72x72",
      "type": "image/png"
    },
    {
      "src": "/images/icons/icon-96x96.png",
      "sizes": "96x96",
      "type": "image/png"
    }
  ]
}

```

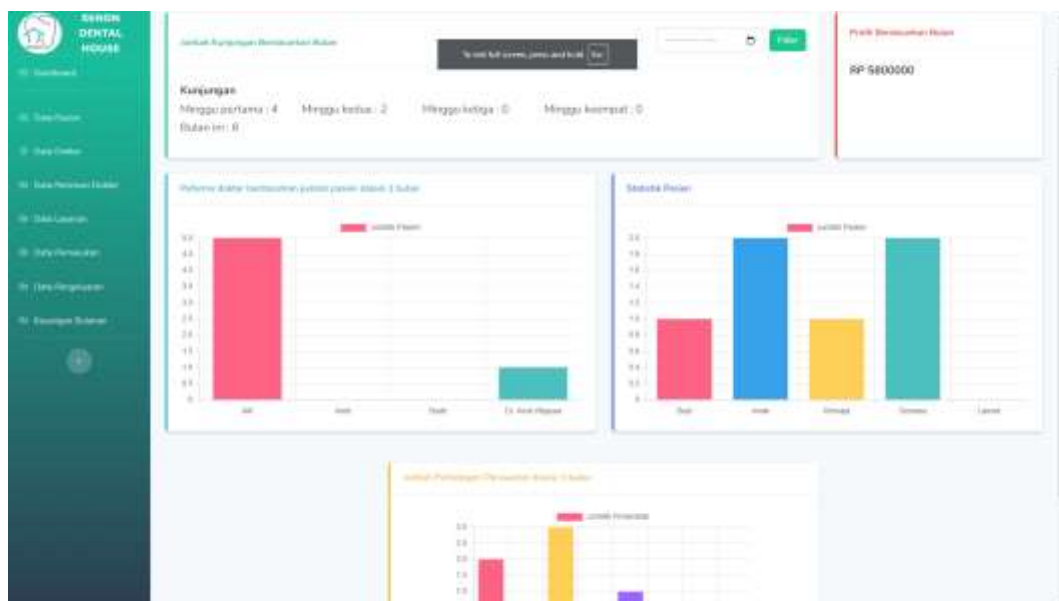
Gambar 5.10 Kode manifest.json

### 6.1.2 Implementasi Antarmuka sistem

Implementasi antarmuka sistem informasi manajemen klinik ini dilakukan menggunakan template engine Laravel blade yang berisikan HTML(HyperText Markup Language) , CCS(Cascading style sheets) serta javascript. Pada bagian ini akan dijelaskan implementasi antarmuka untuk tampilan dashboard owner, dashboard admin, data reservasi dari admin dan data reservasi dari pasien. Implementasi antarmuka lainnya dapat dilihat pada lampiran.

#### 6.1.2.1 Implementasi Antarmuka Halaman Dashboard Owner

Halaman dashboard owner adalah halaman yang menyediakan visualisasi dari data yang ada di klinik. Halaman ini menyediakan informasi profit klinik, visualisasi berupa jumlah kunjungan dalam satu bulan serta terdapat jumlah kunjungan perminggunya pada bulan itu. Disamping itu juga terdapat visualisasi bar chart performa dokter berdasarkan jumlah kunjungan yang diterima oleh dokter , visualisasi jumlah dokter berdasarkan kategori umurnya dan visualisasi jumlah per kategori perawatan yang dilakukan sebanyak perawatan yang sudah dilakukan. Data data yang ditampilkan ini juga bisa ditampilkan berdasarkan bulan yang ingin kita pilih. Hasil dari implementasi halaman dashboard owner ini dapat dilihat pada gambar 5.11 berikut.



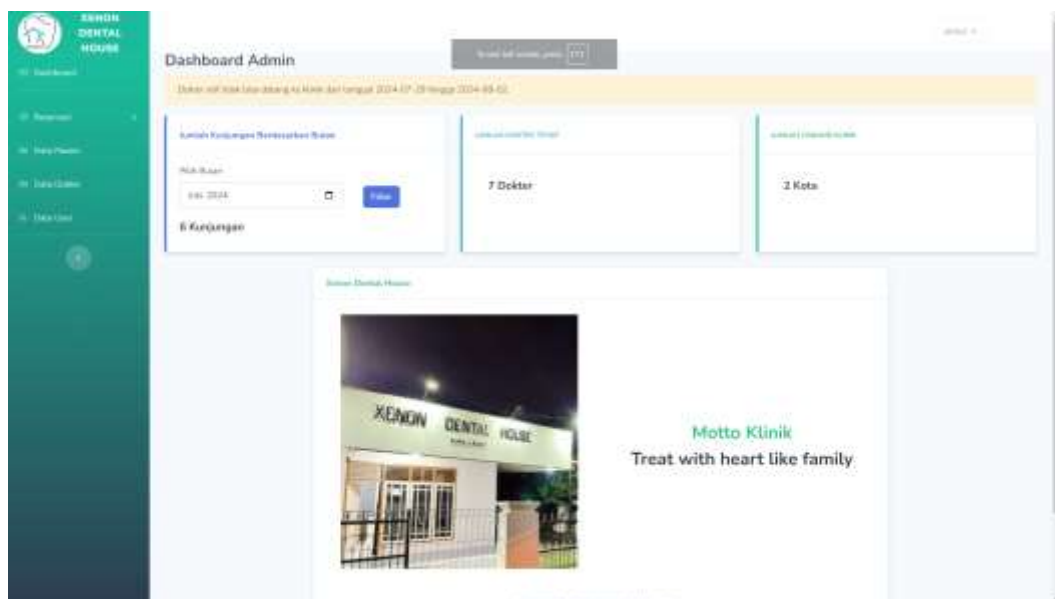
Gambar 5.11 Implementasi Antarmuka Halaman Dashboard Owner

### 6.1.2.2 Implementasi Antarmuka Halaman Dashboard Admin

Halaman dashboard admin adalah halaman pertama yang akan ditampilkan setelah admin login. Halaman ini berisikan informasi terkait jumlah kunjungan pasien dalam bulan ini, jumlah dokter yang bekerja di klinik gigi dan jumlah cabang yang dimiliki oleh klinik gigi. Pada halaman ini juga ditampilkan motto dari klinik gigi xenon dental house. Data data ini juga bisa difilter berdasarkan bulan bulan yang ada sesuai dengan keinginan admin. Hasil dari implementasi ini dapat dilihat pada gambar 5.12 berikut.

### 6.1.2.3 Implementasi Antarmuka Halaman Data Reservasi Admin

Halaman data reservasi admin adalah halaman yang berisikan data inputan reservasi yang telah berhasil dilakukan oleh admin atau data reservasi yang diajukan oleh pasien dan telah disetujui oleh admin. Pada halaman ini ditampilkan data reservasi dimulai dari nama pasien, lokasi perawatan, nama dokter yang akan melakukan praktek, tanggal, nomor hp, jam dan rekam medik dari pasien. Dari halaman ini admin bisa pindah ke berbagai halaman yang diinginkan seperti halaman tambah, edit data reservasi, halaman rekam medik, serta dapat menghapus data reservasi jika diperlukan. Hasil implementasi ini dapat dilihat pada gambar 5.13 berikut.



Gambar 5.12 Implementasi Antarmuka Dashboard Admin

No	Nama Pasien	Lokasi Perawatan	Nama Dokter	Tanggal	Nomor HP	Jam	Status	Aksi
1	Rudi Santoso	Paling	Adi	2024-10-19	800017702747	10:00:00	Siapa yang	Aksi Tolak
2	Rudi Santoso	Paling	Adi	2024-10-07	800017702747	10:00:00	Siapa yang	Aksi Tolak
3	Adi Santoso	Paling	Adi	2024-10-07	800017702747	10:00:00	Siapa yang	Aksi Tolak
4	Adi Santoso	Paling	Adi	2024-10-07	800017702747	10:00:00	Siapa yang	Aksi Tolak
5	Adi Santoso	Paling	Dr. Andi Wijaya	2024-08-05	800017702747	10:00:00	Siapa yang	Aksi Tolak
6	Adi Santoso	Paling	Dr. Andi Wijaya	2024-08-05	800017702747	10:00:00	Siapa yang	Aksi Tolak
7	Adi Santoso	Paling	Adi	2024-07-20	800017702747	10:00:00	Siapa yang	Aksi Tolak
8	Dr. Andi Wijaya	Paling	Adi	2024-07-15	800017702747	11:00:00	Siapa yang	Aksi Tolak

Showing 1 to 8 of 8 entries

Tambah Reservasi Pasien Baru Tampilkan Reservasi Pasien Lama

Gambar 5.13 Implementasi Antarmuka Reservasi dari admin

#### 6.1.2.4 Implementasi Antarmuka Halaman Data Reservasi dari Pasien

Halaman ini merupakan halaman yang menampilkan data reservasi yang diajukan oleh pasien. Data ini berisikan data nama pasien, lokasi, nama dokter, tanggal dan jam. Pada halaman ini admin dapat melakukan penolakan atau penerimaan pada reservasi yang diajukan pasien. Hasil dari implementasi ini dapat dilihat pada gambar 5.14 berikut

No	Nama Pasien	Lokasi Perawatan	Nama Dokter	Tanggal	Jam	Aksi
1	Rudi Santoso	Paling	Adi	2024-07-20	10:00:00	Aksi Tolak

Showing 1 to 1 of 1 entries

Tambah Reservasi Pasien Baru Tampilkan Reservasi Pasien Lama

Gambar 5.14 Implementasi Antarmuka Reservasi Pasien

## 6.2 Pengujian Sistem

Pengujian sistem bertujuan untuk memastikan bahwa aplikasi yang dikembangkan sesuai dengan desain sistem yang telah direncanakan, memastikan hasil yang dirproses dari sistem sama dengan hasil yang diharapkan. Pengujian aplikasi dilakukan menggunakan metode black box. Pengujian sistem dengan metode black box dilakukan dengan membandingkan input dan output yang diharapkan pada setiap fungsionalitas tanpa memperhatikan proses internal dan kinerja aplikasi. Pengujian dilakukan pada semua aktor yang terlibat pada sistem yaitu admin, dokter, owner dan pasien yang menghasilkan hasil pengujian. Pengujian ini lebih lengkapnya dapat dilihat pada lampiran.

### 6.2.1 Fokus Pengujian

Pengujian aplikasi ini berfokus pada penggunaan data uji yang diambil dari aplikasi yang telah dikembangkan. Dalam pengujian ini, terdapat dua puluh lima item yang akan dievaluasi. Fokus pengujian dapat dilihat pada tabel 5.1 dibawah ini.

Tabel 5.1 Pengujian Aplikasi

No	Item Uji	Jenis Aplikasi	Pengguna	Detail Pengujian
1	Authentikasi dan Pengelolaan Akun	Website	Semua User	Login, ganti password, logout, ganti informasi profil, menghapus akun pribadi
2	Lihat Dashboard	Website	Admin, Owner, Dokter	Lihat, Filter
3	Konfirmasi reservasi dari Pasien	Website	Admin	Lihat, Tolak, terima
4	Reservasi pasien	Website	Admin	Lihat, tambah, edit, hapus
5	Mengelola data pasien	Website	Admin	Lihat, edit, hapus

Tabel 5.1 Pengujian Aplikasi(lanjutan)

6	Mengelola data rekam medik	<i>Website</i>	Admin	Lihat, edit
7	Mengelola data dokter, jadwal dokter	<i>Website</i>	Admin	Lihat, tambah, edit, hapus
8	Mengelola data user	<i>Website</i>	Admin	Lihat, tambah, edit, hapus
9	Registrasi Akun	<i>Website</i>	Pasien	Registrasi akun baru
10	Reservasi klinik	<i>Website</i>	Pasien	Tambah, Edit
11	Melihat Riwayat reservasi	<i>Website</i>	Pasien	Lihat
12	Lihat Jadwal Praktek	<i>Website</i>	Dokter	Lihat
13	Mengelola data pasien	<i>Website</i>	Dokter	Lihat, edit, hapus
14	Mengajukan permintaan izin	<i>Website</i>	Dokter	Lihat, tambah, edit, hapus
15	Melihat data profit klinik	<i>Website</i>	Owner	Lihat, Filter
16	Melihat jumlah kunjungan pasien	<i>Website</i>	Owner	Lihat, Filter
17	Melihat jumlah penanganan perawatan berdasarkan kategori	<i>Website</i>	Owner	Lihat, Filter
18	Melihat performa dokter berdasarkan jumlah pasien	<i>Website</i>	Owner	Lihat, Filter
19	Melihat data pasien berdasarkan umur	<i>Website</i>	Owner	Lihat, Filter
20	Melihat Data dokter	<i>Website</i>	Owner	Lihat
21	Melihat Data Pasien	<i>Website</i>	Owner	Lihat
22	Melihat Data Pemasukan, keuangan	<i>Website</i>	Owner	Lihat
23	Memutuskan perizinan dokter	<i>Website</i>	Owner	Terima, Tolak



Tabel 5.1 Pengujian Aplikasi(lanjutan)

24	Mengelola pencatatan pengeluaran klinik	Website	Owner	Lihat, tambah, edit, hapus
25	Mengelola harga perawatan	Website	Owner	Lihat, tambah, edit, hapus

### 6.2.2 Kasus Hasil Pengujian

Kasus hasil pengujian adalah pembahasan mengenai kasus dan hasil pengujian terhadap fungsionalitas aplikasi yang telah dilakukan berdasarkan fokus pengujian yang tercantum pada Tabel 5.1. Dalam sub bab ini akan dijelaskan mengenai kasus hasil pengujian yang mencakup Input Reservasi dari Admin, Registrasi Akun Pasien, Admin menerima pengajuan reservasi dari pasien

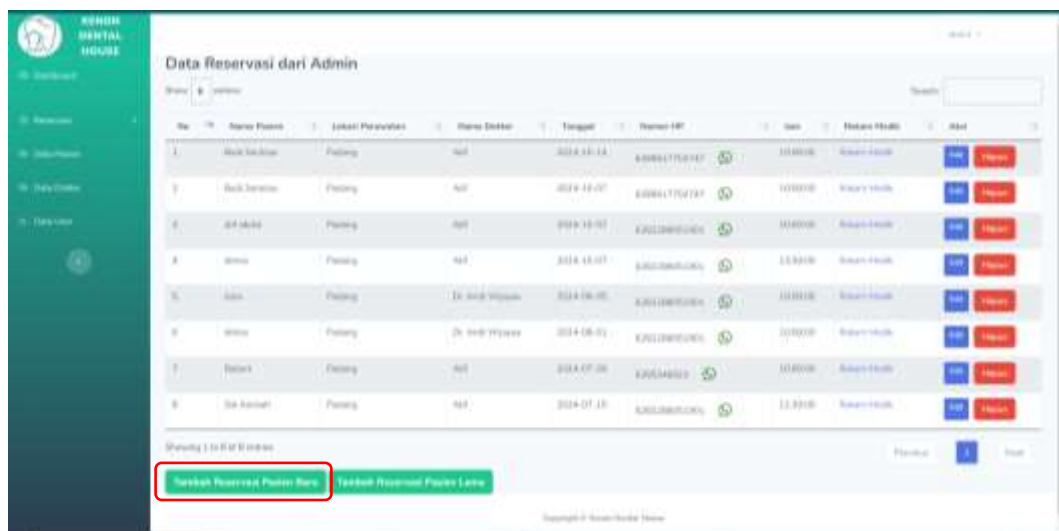
#### 6.2.2.1 Input Reservasi Pasien baru dari Admin

Pada pengujian ini admin akan menginputkan reservasi untuk pasien baru ketika pasien datang klinik Xenon Dental house. Pada kasus kali ini pasien bernama Andy meminta reservasi pada tanggal 18 Juli 2024. Hasil dari pengujian ini dapat dilihat pada tabel 5.2 dibawah ini :

Tabel 5.2 Pengujian kondisi benar input reservasi dari admin

Kasus dan Hasil Uji (benar)	
Data masukan	Semua data yang dibutuhkan untuk reservasi seperti detail reservasi dan data pribadi pasien
Hasil yang diharapkan	Data tersimpan resever system menampilkan list reservasi yang sudah ditambahkan
Pengamatan	Sistem berhasil menyimpan dan menampilkan list reservasi yang sudah ditambahkan beserta notifikasi berhasil
Hasil	Sesuai

Pengujian diawali dengan admin membuka menu reservasi admin pada *website*. Setelah berada pada halaman reservasi, admin menekan tombol tambahkan reservasi pasien baru. Admin akan diarahkan pada halaman form tambah reservasi. Admin perlu mengisi semua form yang diperlukan untuk reservasi, setelah mengisi semua form maka admin menekan tombol submit. Jika berhasil akan muncul notifikasi berhasil dan admin akan diarahkan kepada halaman list reservasi. Data yang baru diinputkan akan muncul pada list reservasi. Pengujian ini dapat dilihat ada gambar 5.15, 5.16, dan 5.17 berikut.



No	Nama Pasien	Jenis Perawatan	Status Dokter	Tanggal	Reservasi ID	Waktu	Status Resisi	Aksi
1	Andi Setiawan	Pemang	Ada	2024-10-14	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
2	Budi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
3	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
4	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
5	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
6	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
7	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>
8	Andi Setiawan	Pemang	Ada	2024-10-07	600041770747	10:00:00	Reservasi Baru	<a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 8 of 8 entries

[Tambah Reservasi Pasien Baru](#) [Tambah Reservasi Pasien Lama](#)

Gambar 5.15 Tampilan awal reservasi dari admin



**Form Tambah Reservasi**

Nama:

Alamat:

Telepon:

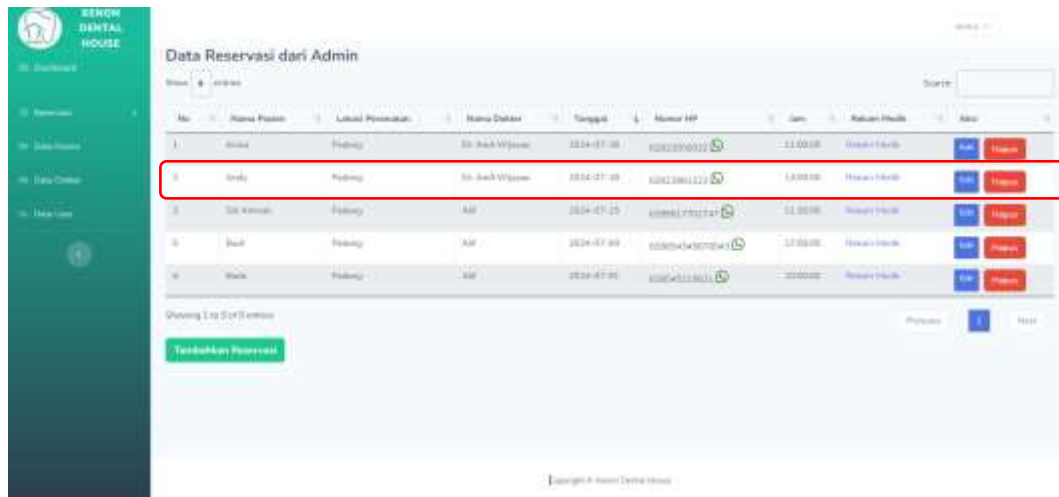
Email:

Tanggal:

Waktu:

Lokasi:

Gambar 5.16 Tampilan Form tambah reservasi pasien baru



Gambar 5.17 Tampilan reservasi setelah data berhasil ditambahkan

Selanjutnya adalah pengujian alternatif, pengujian alternatif adalah pengujian untuk kasus yang berbeda dibandingkan kasus benar. Pengujian alternatif untuk fungsional input reservasi terjadi ketika admin tidak mengisi semua form yang ada di reservasi. Hasil pengujian ini dapat dilihat pada tabel 5.3

Tabel 5.3 Pengujian alternatif reservasi dari admin

Kasus dan Hasil Uji (alternatif)	
Data masukan	Mengisikan hanya dari form reservasi
Hasil yang diharapkan	Akan muncul notifikasi untuk mengisi form yang dikosongkan
Pengamatan	Notifikasi form yang dikosongkan menunjukkan untuk mengisi form yang kosong
Hasil	Sesuai

Gambar 5.18 Tampilan notifikasi saat input form reservasi tidak diisi

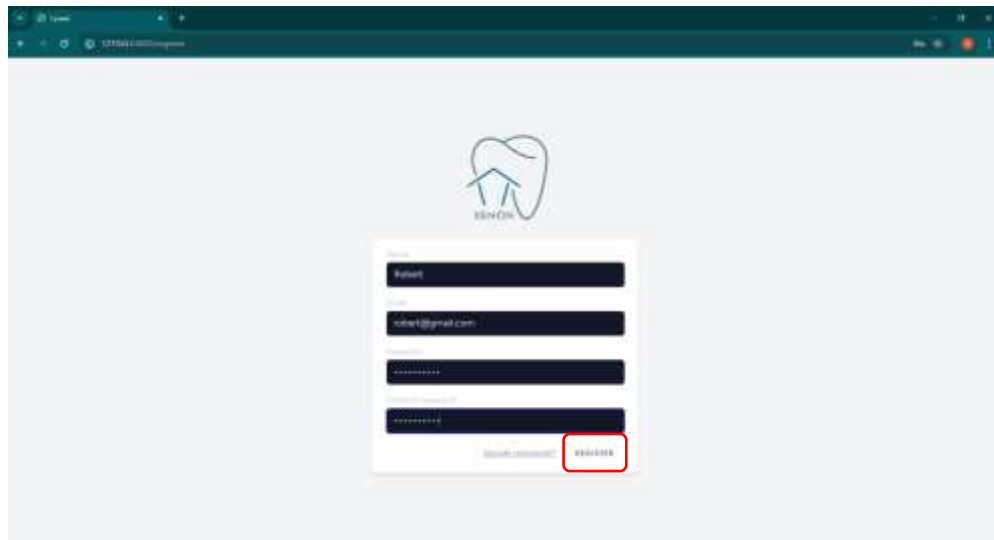
### 6.2.2.2 Registrasi Akun pasien

Pada pengujian ini pasien akan melakukan registrasi akun baru agar bisa melakukan reservasi pada *website*. Pada kasus kali ini pasien melakukan registrasi dengan mengisi pada halaman registrasi. Hasil dari pengujian ini dapat dilihat pada tabel 5.4 dibawah ini :

Tabel 5.4 Pengujian kondisi benar registrasi pasien

Kasus dan Hasil Uji (benar)	
Data masukan	Semua data yang dibutuhkan untuk registrasi, seperti nama, email, password
Hasil yang diharapkan	Data tersimpan dan system akan menampilkan landing page untuk akun pasien yang sudah login
Pengamatan	Sistem berhasil menyimpan data dan berhasil mengarahkan ke halaman landing page
Hasil	Sesuai

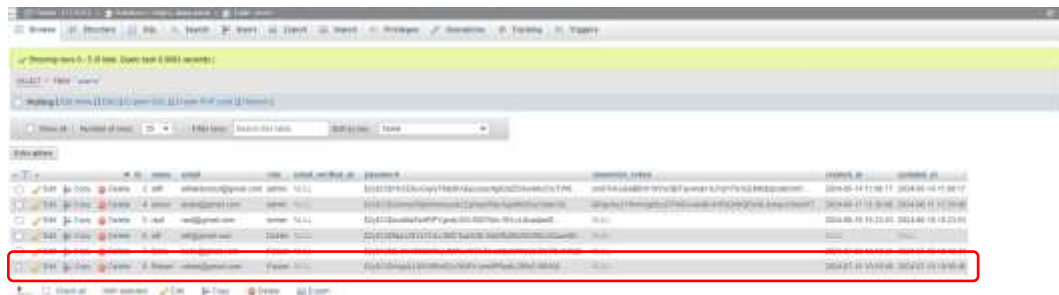
Pengujian diawali dengan pasien masuk pada halaman registrasi akun pada *website*. Setelah berada pada halaman registrasi, pasien mengisi data data yang diperlukan untuk membuat akun baru. Setelah mengisi data data tersebut pasien menekan tombol register. Ketika berhasil maka pasien akan diarahkan ke tampilan landing page serta pasien bisa mengakses menu reservasi pada landing page. Data user akan terlihat bertambah di tabel user database *website*. Hasil pengujian ini dapat dilihat pada gambar 5.19, 5.20, dan 5.21.

A screenshot of a web browser showing a registration form for 'XENON DENTAL HOUSE'. The form is centered on a light blue background. It includes fields for 'Nama' (Name), 'Email', 'Password', and 'Confirm Password'. A red box highlights the 'REGISTER' button at the bottom right of the form. Above the form is a logo featuring a tooth and the text 'XENON'.

Gambar 5.19 Tampilan form registrasi akun



Gambar 5.20 Tampilan Setelah pasien berhasil register

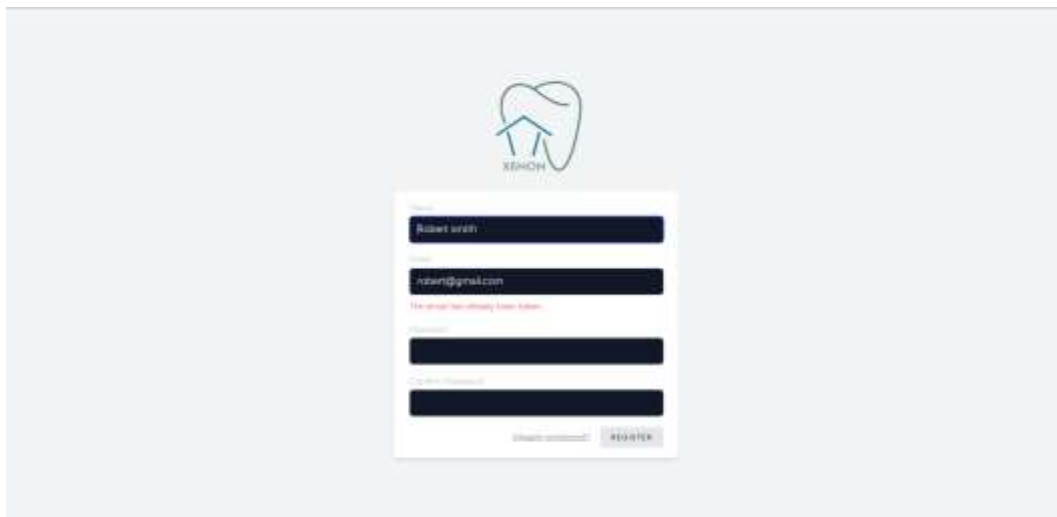


Gambar 5.21 Penambahan data user setelah registrasi akun pasien dilakukan

Selanjutnya pengujian alternatif. Kasus untuk pengujian alternatif adalah ketika user memasukkan email yang sudah terdaftar pada *website*. User akan diminta memasukkan email lain dikarenakan email tersebut sudah terdaftar. Hasil dari pengujian ini dapat dilihat pada tabel 5.5 dan gambar 5.20

Tabel 5.5 Pengujian alternatif registrasi akun pasien

Kasus dan Hasil Uji (alternatif)	
Data masukan	Mengisikan email yang sudah terdaftar di <i>website</i>
Hasil yang diharapkan	Akan muncul notifikasi yang memberitahu bahwa email yang diinputkan sudah didaftarkan.
Pengamatan	Notifikasi pemberitahuan muncul yang menginformasikan bahwa email sudah didaftarkan
Hasil	Sesuai



Gambar 5.22 Notifikasi saat mendaftarkan akun email yang sudah ada

### 6.2.2.3 Admin Menerima pengajuan reservasi dari pasien

Pada pengujian ini pasien akan mengajukan reservasi melalui akunnya. Setelah itu admin akan memeriksa pengajuan ini dan menyetujui/menerima reservasi ini melalui akun admin. Pada kasus ini pasien bernama Robert dan user admin bernama abdul. Hasil dari pengujian ini dapat dilihat pada tabel 5.6 dibawah ini :

Tabel 5.6 Pengujian kondisi benar reservasi dari pasien

<b>Kasus dan Hasil Uji (benar)</b>	
Data masukan	Reservasi dari pasien serta persetujuan dari admin
Hasil yang diharapkan	Data tersimpan dan reservasi yang diajukan akan berubah statusnya menjadi diterima.
Pengamatan	Sistem berhasil menyimpan data dan berhasil mengubah status dari reservasi yang diajukan
Hasil	Sesuai

Pada pengujian ini user pasien perlu mengajukan reservasi terlebih dahulu dengan cara mengisi form tambah reservasi pada *website*. Setelah selesai diinputkan maka yang data akan muncul pada reservasi dengan status belum diproses dan muncul pada draft reservasi dari pasien. Setelah itu admin akan menerima pengajuan yang membuat status reservasi pada pasien akan berubah menjadi diterima dan data reservasi pada admin akan dipindahkan ke bagian reservasi dari admin. Proses ini dapat dilihat pada gambar 5.23, 5.24, 5.25, 5.26.

The screenshot shows the 'Form Tambah Reservasi' interface. The left sidebar contains the 'SEHON DENTAL HOUSE' logo and navigation links. The main content area is a form with the following fields: 'Nama' (Name), 'Email', 'No Telp' (Phone Number), 'Alamat' (Address), and a 'Status' dropdown menu. The 'Status' dropdown is currently set to 'Belum diproses' (Not processed).

Gambar 5.23 Tampilan form saat pasien melakukan reservasi

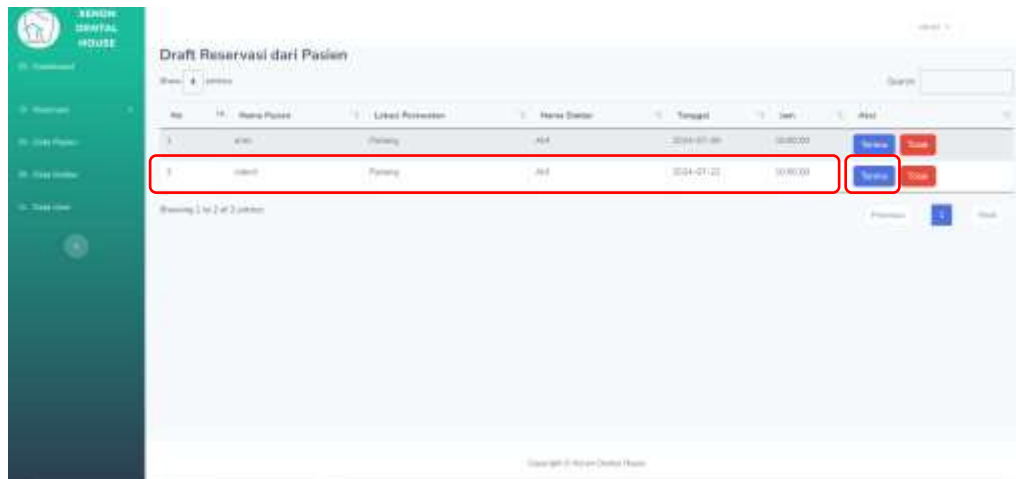
The screenshot shows the 'History Reservasi' table. The table has columns for 'No', 'Nama Pasien', 'Lokasi Prosedur', 'Status', 'Tanggal', 'Jam', and 'Status'. The data row shows a reservation for 'Rendi' at 'Fidury' on '2024-07-22' at '10:00:00' with a status of 'Belum diproses'.

No	Nama Pasien	Lokasi Prosedur	Status	Tanggal	Jam	Status
1	Rendi	Fidury	Belum diproses	2024-07-22	10:00:00	Belum diproses

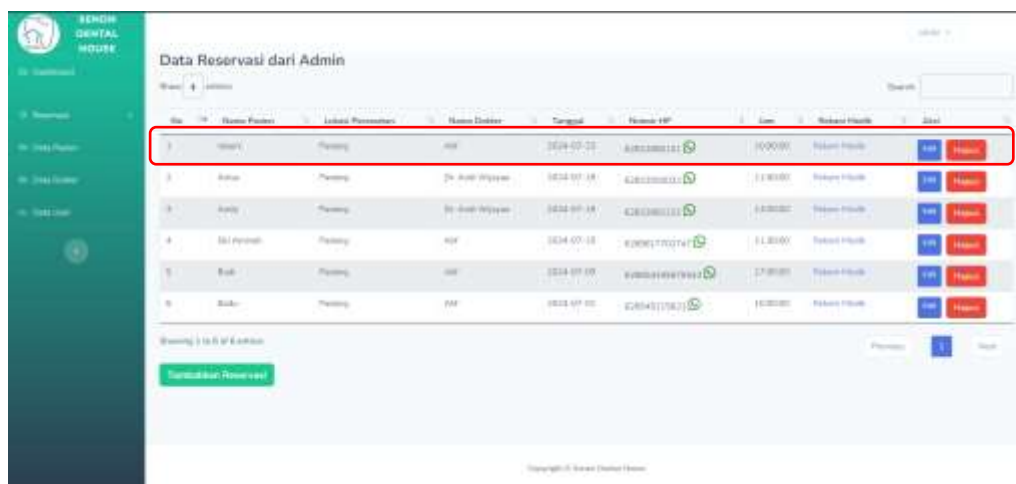
Showing 1 of 1 records

Gambar 5.24 Tampilan history pasien selesai melakukan reservasi

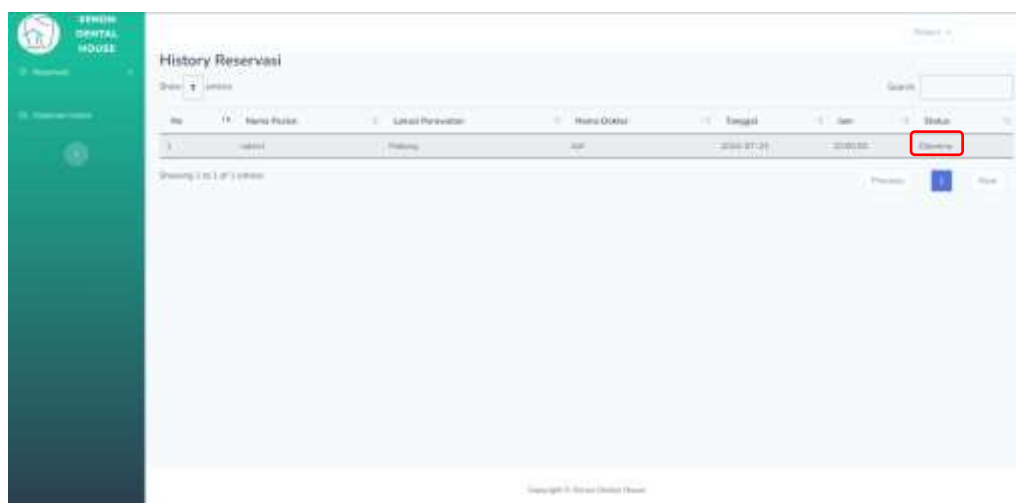




Gambar 5.25 Tampilan user admin setelah pasien selesai melakukan reservasi



Gambar 5.26 Tampilan data reservasi admin setelah reservasi pasien diterima



Gambar 5.27 Tampilan *history* reservasi pasien setelah reservasi pasien diterima

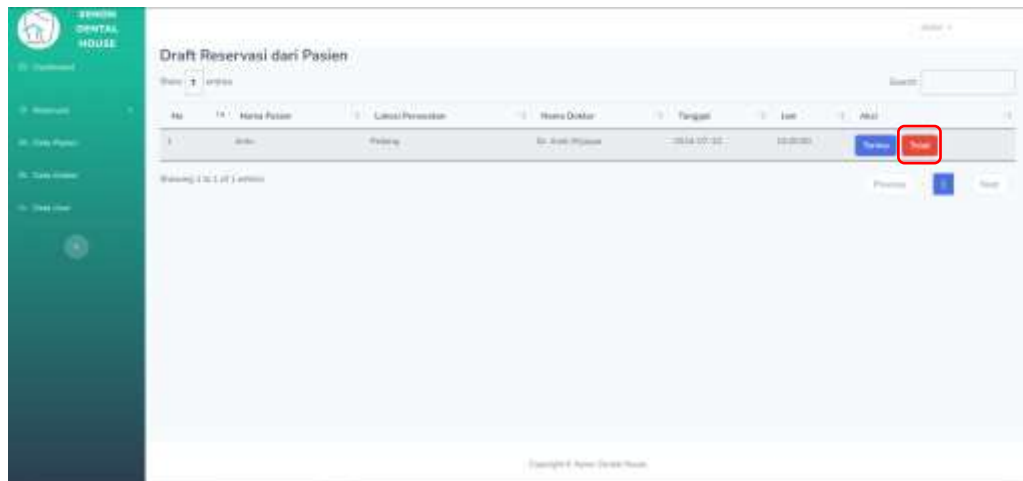
Selanjutnya pengujian alternatif. Pada pengujian ini admin juga bisa menolak pengajuan dari pasien dengan cara menekan tombol tolak. Status reservasi pada akun pasien akan menjadi ditolak serta data akan hilang dari tampilan admin. Hasil dari pengujian ini dapat dilihat pada tabel 5.7 dan gambar 5.26 , 5.27, 5.28.

Tabel 5.7 Pengujian alternatif reservasi dari pasien

Kasus dan Hasil Uji (alternatif)	
Data masukan	Dokter menolak pengajuan reservasi dari pasien
Hasil yang diharapkan	Status reservasi dari pasien akan berubah menjadi ditolak serta data akan hilang dari tampilan admin
Pengamatan	Status reservasi pada pasien adalah ditolak dan data sudah tidak ada pada tampilan admin
Hasil	Sesuai



Gambar 5.28 Tampilan history reservasi pasien setelah melakukan reservasi



Gambar 5.29 Tampilan reservasi pasien setelah melakukan reservasi pada admin



Gambar 5.30 Tampilan histori reservasi pasien setelah ditolak admin

#### 6.2.2.4 Pengujian mode *offline PWA* pada *website*

Pada pengujian ini dilakukan uji coba mode offline dari *website* yang menggunakan pendekatan *PWA(Progressive Web App)*. Pengujian ini dilakukan ketika kondisi server mati atau user tidak bisa mengakses server dari *website*.

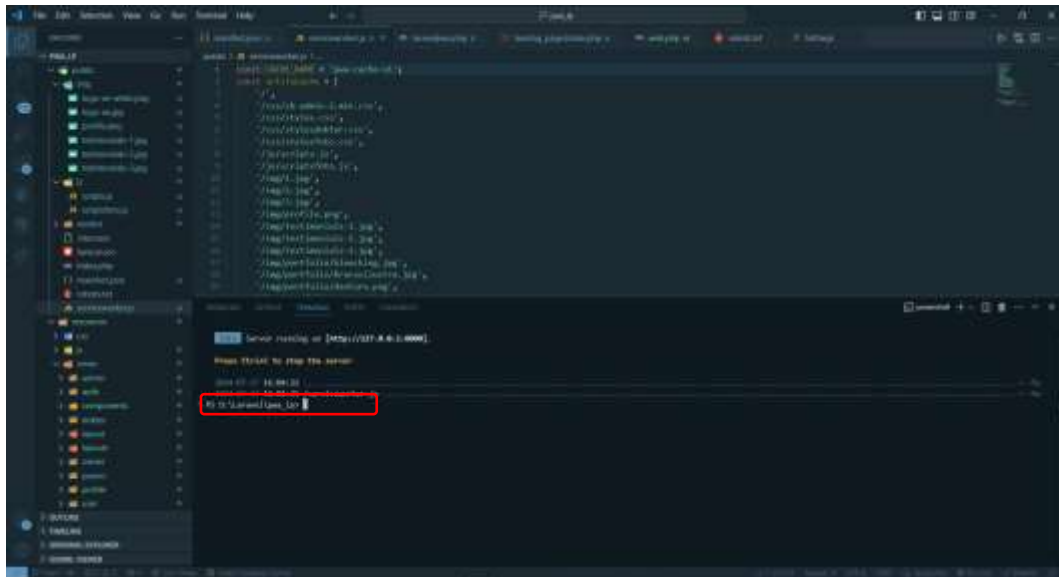
Tabel 5.8 Pengujian mode offline pada *website*

Kasus dan Hasil Uji (benar)	
Kondisi pengujian	Server dari <i>website</i> akan dimatikan
Hasil yang diharapkan	<i>Website</i> tetap bisa menampilkan landing page dari <i>website</i> dengan cara menyimpan cache dari <i>website</i> .
Pengamatan	Sistem berhasil menampilkan <i>website</i> dalam kondisi offline walaupun setelah dimuat ulang.
Hasil	Sesuai

Pada pengujian ini pertama user menampilkan kondisi ketika *website* ketika server masih aktif. Setelah itu dilakukan uji coba dengan mematikan server Laravel serta mengaktifkan fitur offline pada chrome. Setelah itu *website* akan dimuat ulang dalam kondisi server tidak aktif. Hasilnya *website* akan tetap bisa menampilkan landing page dari *website* dengan fungsional yang terbatas.



Gambar 5.31 Tampilan landing page saat server aktif



Gambar 5.32 Tampilan saat server dimatikan



Gambar 5.33 Tampilan landing page tetap muncul setelah server mati

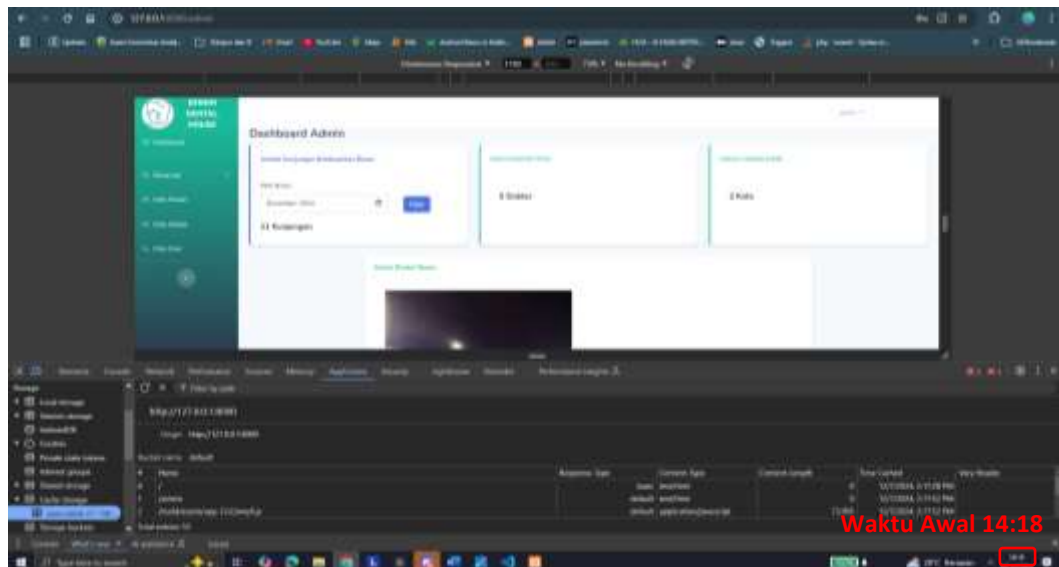
#### 6.2.2.5 Pengujian Lama Kadaluarsa Cache untuk Akses *PWA offline*

Pada pengujian ini dilakukan uji coba kadaluarsa dari *cache* yang disimpan di *browser*. Ketika *cache* kadaluarsa, maka user tidak akan bisa mengakses halaman secara offline lagi. Pada pengujian ini terdapat 2 kondisi, kondisi pertama lama kadaluarsa *cache* tidak akan diatur, yang mana membuat *cache* tidak akan dihapus kecuali ada tindakan menghapus dari user. Kondisi kedua, lama kadaluarsa dari *cache* akan diatur. *Cache* akan dihapus ketika website sudah tidak terhubung ke server selama 1 jam.

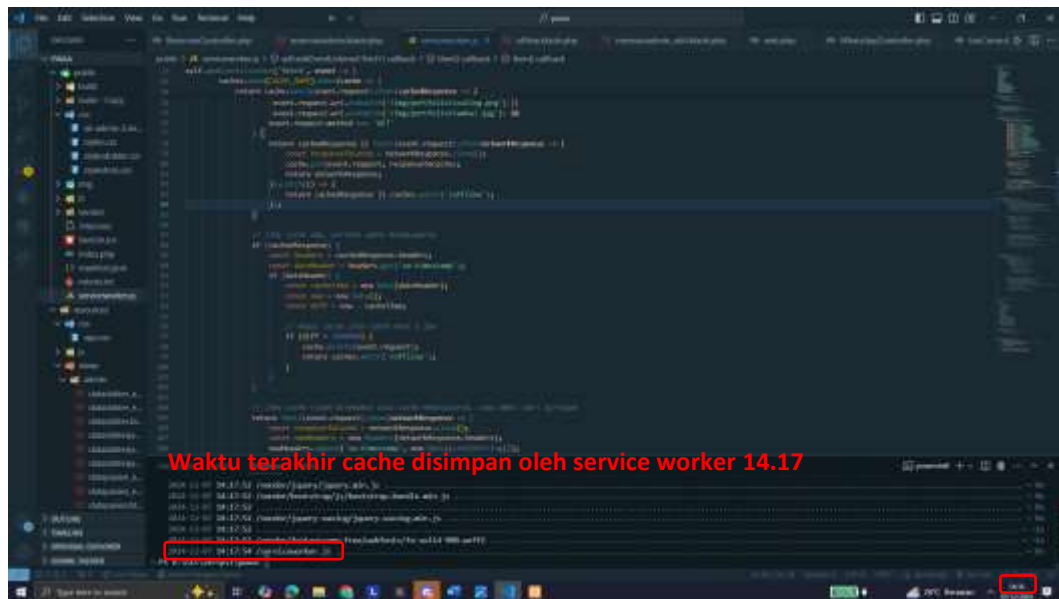
Tabel 5.9 Pengujian Sistem Membatasi akses ke cache jika lebih 1 jam

Kasus dan Hasil Uji (Batas akses cache dibatasi 1 jam)	
Kondisi pengujian	Website tidak dihubungkan ke server selama lebih dari 1 jam.
Hasil yang diharapkan	Website akan menampilkan pemberitahuan pembatasan akses
Pengamatan	Sistem berhasil menampilkan <i>website</i> selama 1 jam awal, dan membatasi akses ketika lebih dari 1 jam
Hasil	Sesuai

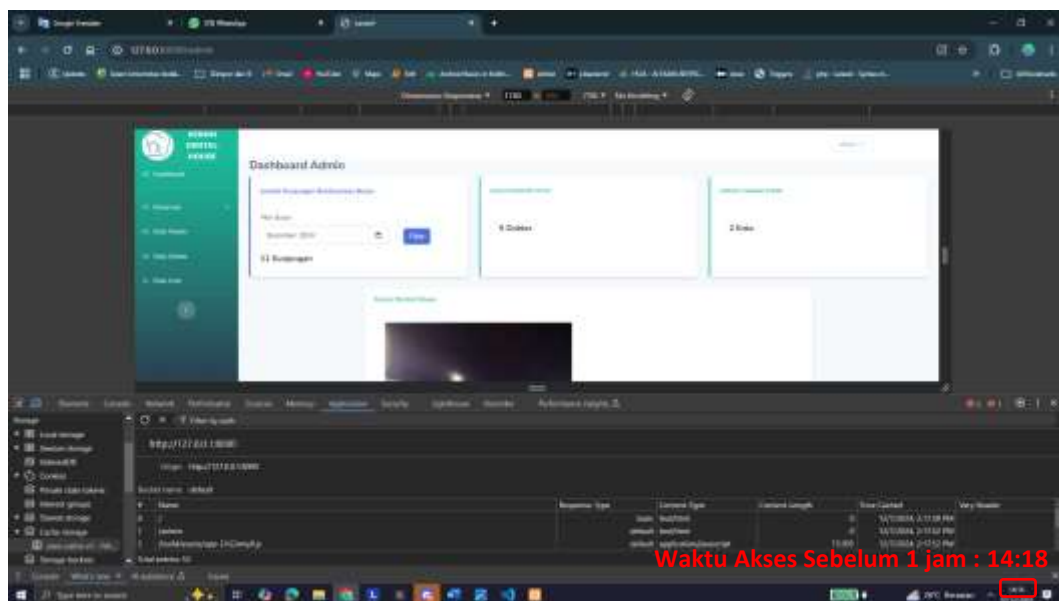
Pada pengujian pertama, website akan ditampilkan ketika dalam kondisi terhubung ke server. Setelah itu server akan dimatikan dan website akan dimuat ulang, karena jarak waktu website terputus dengan server masih kurang dari 1 jam, maka website masih bisa diakses secara offline. Lalu website akan dibiarkan selama lebih dari 1 jam, dan setelah lebih dari 1 jam maka website akan dimuat ulang. Hasilnya akan menampilkan bahwa akses website telah dibatasi karena website sudah tidak terhubung ke server lebih dari 1 jam.



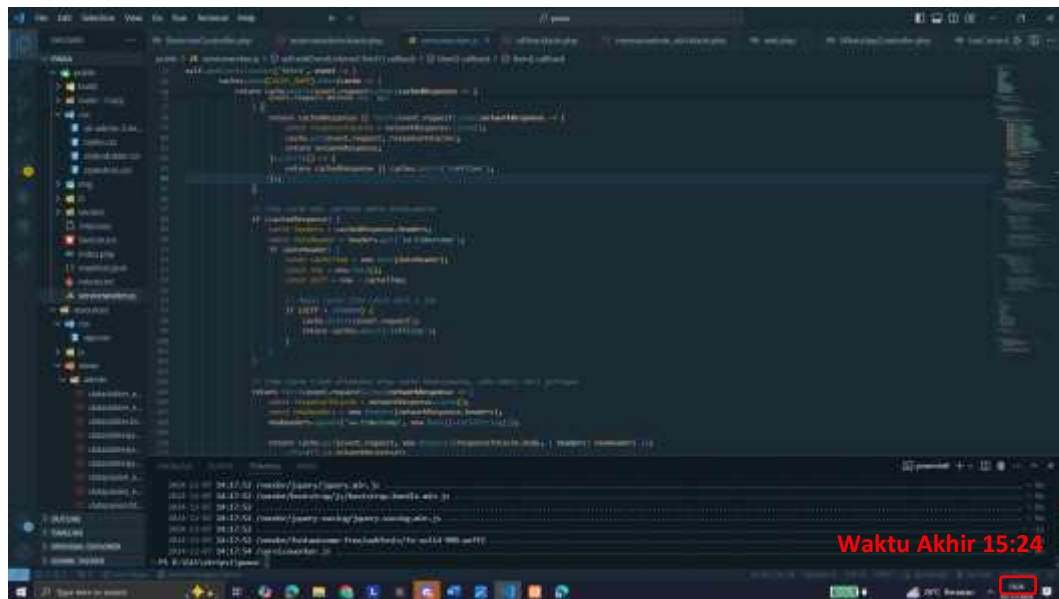
Gambar 5.34 Tampilan Web sebelum server dimatikan



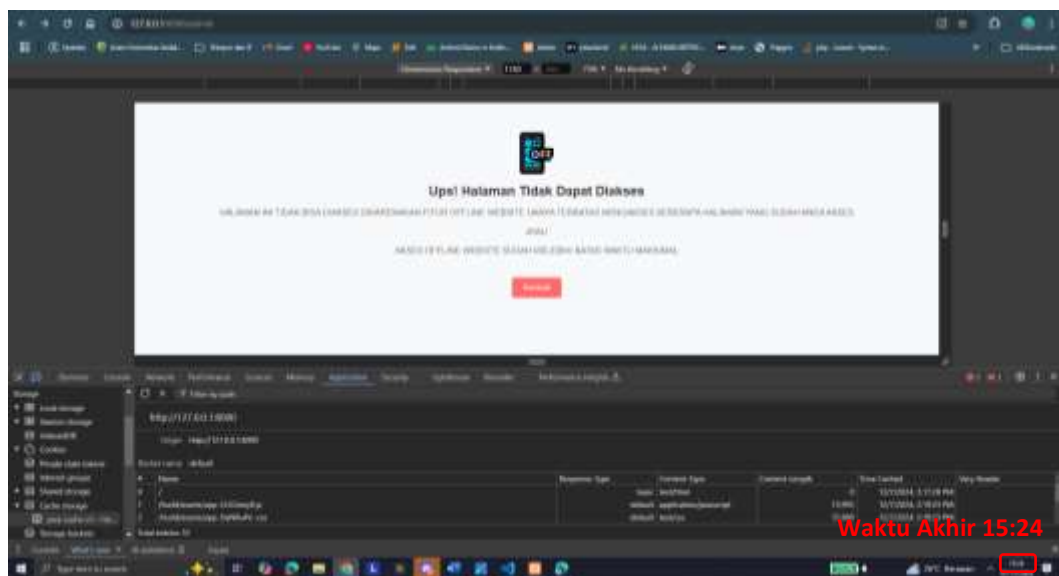
Gambar 5.35 Tampilan server dimatikan



Gambar 5.36 Tampilan web ketika server dimatikan sebelum lewat 1 jam



Gambar 5.37 Tampilan server setelah lewat 1 jam



Gambar 5.38 Tampilan website setelah server dimatikan lewat 1 jam

Pada pengujian kedua dengan kondisi waktu kadaluarsa cache tidak dibatasi, awalnya website akan ditampilkan ketika dalam kondisi terhubung ke server. Seperti perlakuan pada pengujian pertama, server akan dimatikan dan website akan dimuat ulang ketika jarak waktu website terputus dengan server masih kurang dari 1 jam dan lebih dari 1 jam. Hasilnya akan menampilkan bahwa akses website tetap bisa diakses sebelum maupun setelah 1 jam website tidak terhubung ke server.

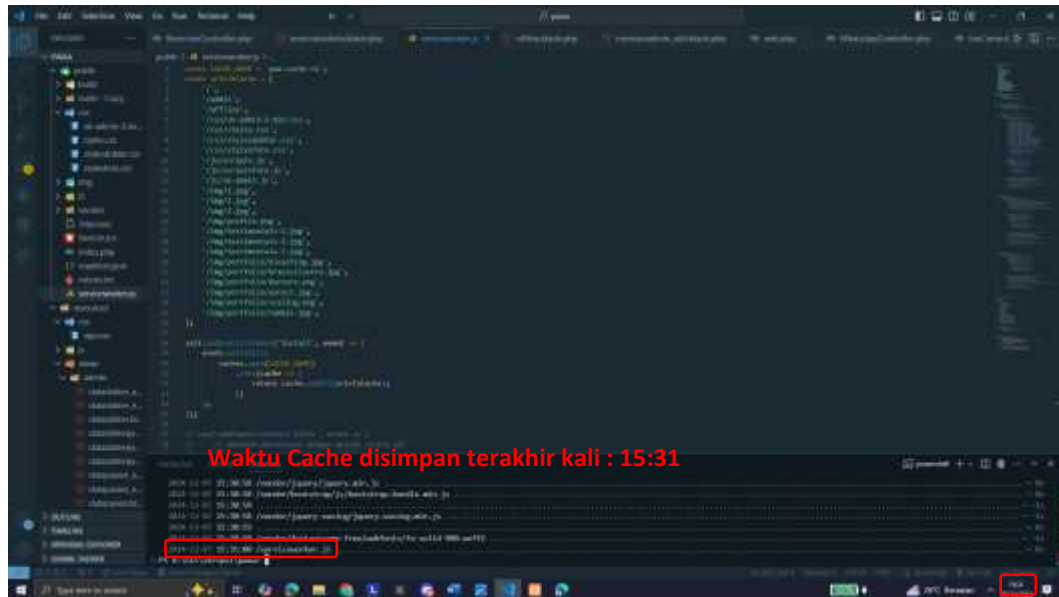


Tabel 5.10 Pengujian Sistem Lama waktu akses cache tidak dibatasi

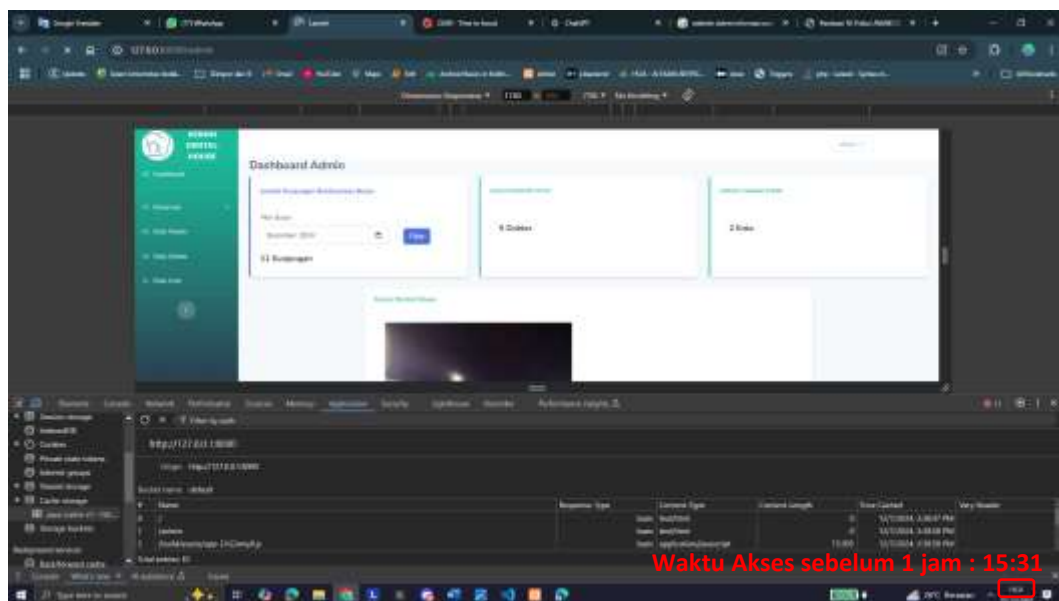
Kasus dan Hasil Uji (Lama waktu akses cache tidak dibatasi)	
Kondisi pengujian	Website tidak dihubungkan ke server selama lebih dari 1 jam.
Hasil yang diharapkan	Website akan tetap bisa diakses secara <i>offline</i> setelah dan sebelum 1 jam terputus dari server
Pengamatan	Sistem berhasil diakses secara <i>offline</i> sebelum dan setelah 1 jam
Hasil	Sesuai



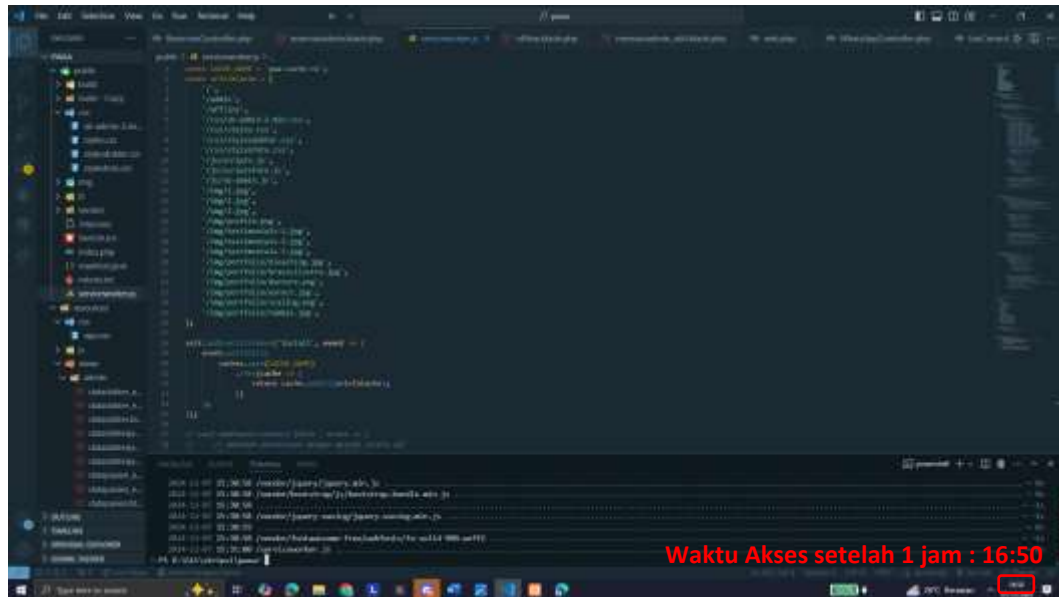
Gambar 5.39 Tampilan website sebelum server dimatikan



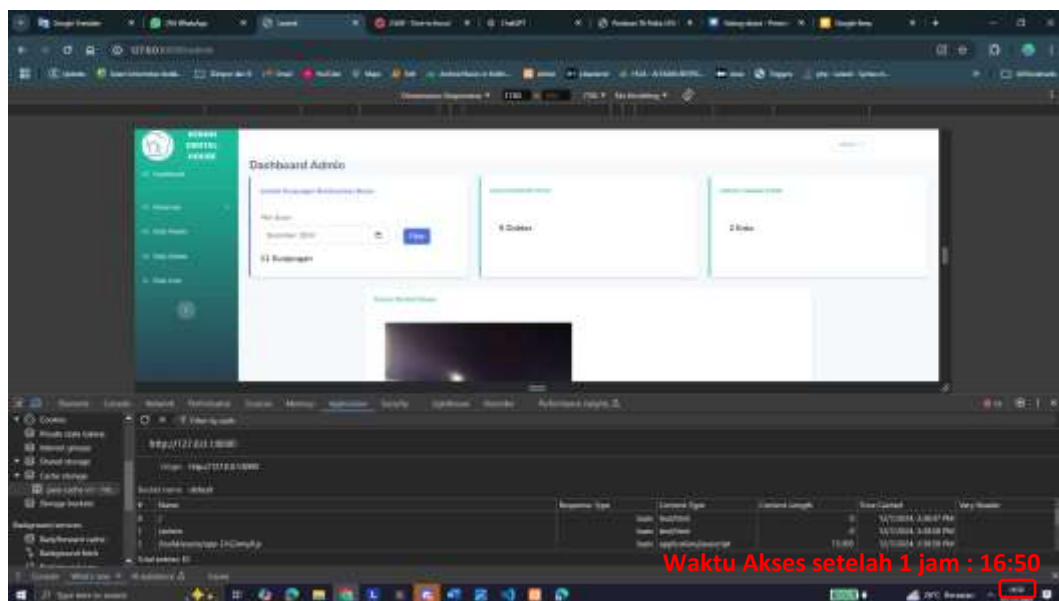
Gambar 5.40 Tampilan server setelah dimatikan



Gambar 5.41 Tampilan website ketika server dimatikan sebelum lewat 1 jam



Gambar 5.42 Tampilan server setelah lewat 1 jam



Gambar 5.43 Tampilan website setelah server dimatikan lewat 1 jam

### 6.2.2.6 Kesimpulan hasil pengujian

Dari hasil pengujian aplikasi web yang telah dilakukan menunjukkan bahwa *website* yang dikembangkan telah sesuai dengan rancangan sistem yang direncanakan. Pengujian menggunakan metode black box mengonfirmasi bahwa semua fungsionalitas beroperasi sesuai harapan, dengan input dan output yang tepat pada setiap item yang diuji. Oleh sebab itu, dapat disimpulkan bahwa aplikasi

sistem informasi manajemen klinik gigi Xenon Dental House telah berjalan sesuai dengan kebutuhan fungsional yang telah dirancang. Secara keseluruhan, aplikasi web memenuhi semua kriteria pengujian yang telah ditetapkan, menunjukkan bahwa aplikasi siap untuk digunakan. Kesimpulan hasil pengujian dapat dilihat pada tabel 5.11 berikut.

Tabel 5.11 Kesimpulan hasil pengujian

No	Item Uji	Jenis Aplikasi	Pengguna	Detail Pengujian	Hasil
1	Authentikasi dan Pengelolaan Akun	<i>Website</i>	Semua User	Login, ganti password, logout, ganti informasi profil, menghapus akun pribadi	Sesuai
2	Lihat Dashboard	<i>Website</i>	Admin, Owner, Dokter	Lihat, Filter	Sesuai
3	Konfirmasi reservasi dari Pasien	<i>Website</i>	Admin	Lihat, Tolak, terima	Sesuai
4	Reservasi pasien	<i>Website</i>	Admin	Lihat, tambah, edit, hapus	Sesuai
5	Mengelola data pasien	<i>Website</i>	Admin	Lihat, edit, hapus	Sesuai
6	Mengelola data rekam medik	<i>Website</i>	Admin	Lihat, edit	Sesuai
7	Mengelola data dokter, jadwal dokter	<i>Website</i>	Admin	Lihat, tambah, edit, hapus	Sesuai
8	Mengelola data user	<i>Website</i>	Admin	Lihat, tambah, edit, hapus	Sesuai
9	Registrasi Akun	<i>Website</i>	Pasien	Registrasi akun baru	Sesuai
10	Reservasi klinik	<i>Website</i>	Pasien	Tambah, Edit	Sesuai
11	Melihat Riwayat reservasi	<i>Website</i>	Pasien	Lihat	Sesuai

Tabel 5.11 Kesimpulan hasil pengujian (lanjutan)

12	Lihat Jadwal Praktek	<i>Website</i>	Dokter	Lihat	Sesuai
13	Mengelola data pasien	<i>Website</i>	Dokter	Lihat, edit, hapus	Sesuai
14	Mengajukan permintaan izin	<i>Website</i>	Dokter	Lihat, tambah, edit, hapus	Sesuai
15	Melihat data profit klinik	<i>Website</i>	Owner	Lihat, Filter	Sesuai
16	Melihat jumlah kunjungan pasien	<i>Website</i>	Owner	Lihat, Filter	Sesuai
17	Melihat jumlah penanganan perawatan berdasarkan kategori	<i>Website</i>	Owner	Lihat, Filter	Sesuai
18	Melihat performa dokter berdasarkan jumlah pasien	<i>Website</i>	Owner	Lihat, Filter	Sesuai
19	Melihat statistik pasien berdasarkan umur	<i>Website</i>	Owner	Lihat, Filter	Sesuai
20	Melihat Data dokter	<i>Website</i>	Owner	Lihat	Sesuai
21	Melihat Data Pasien	<i>Website</i>	Owner	Lihat	Sesuai
22	Melihat data Pemasukan dan keuangan klinik	<i>Website</i>	Owner	Lihat	Sesuai
23	Memutuskan perizinan dokter	<i>Website</i>	Owner	Terima, Tolak	Sesuai
24	Mengelola pencatatan pengeluaran klinik	<i>Website</i>	Owner	Lihat, tambah, edit, hapus	Sesuai
25	Mengelola harga perawatan	<i>Website</i>	Owner	Lihat, tambah, edit, hapus	Sesuai

### 6.2.2.7 Analisa dan Pembahasan Hasil

Sistem informasi yang dikembangkan memiliki beberapa fitur yang dirancang untuk mendukung pengelolaan klinik dan layanan kesehatan dengan lebih efektif dibandingkan sistem yang ada sebelumnya. Berikut analisis perbandingan antara sistem manual yang sebelumnya digunakan dengan sistem berbasis web yang telah dikembangkan.

#### 1. Sentralisasi data dan Terorganisir

Sistem sebelumnya menggunakan berbagai media untuk mencatat data, mulai dari catatan manual hingga reservasi melalui WhatsApp. Tidak adanya satu pusat data yang terintegrasi dapat diatasi dengan dibangunnya website ini dalam mengakses dan memperbarui informasi. Sistem informasi manajemen berbasis web ini memusatkan semua data pasien, dokter, layanan, serta rekam medis dalam satu platform digital yang dapat diakses oleh pihak yang berwenang serta akses *realtime* memungkinkan klinik mengakses data antar cabang pada klinik tanpa memandang lokasi.

#### 2. Keamanan dan Akurasi Data

Pada sistem sebelumnya, keamanan data menjadi perhatian penting karena saat ini data seperti rekam medik masih di simpan dalam bentuk fisik(kertas). Data yang tersimpan dalam bentuk fisik (kertas) atau melalui WhatsApp tidak memiliki kontrol akses yang ketat, sehingga rawan terhadap akses tidak sah atau bahkan kehilangan. Dengan penerapan sistem informasi ini, keamanan informasi lebih terjamin melalui pengaturan hak akses yang terkelola dan enkripsi data. Selain itu, data yang diinput ke dalam sistem memiliki mekanisme validasi yang meminimalkan kesalahan input, sehingga kualitas dan akurasi data lebih terjaga.

### 3. Otomasi Penjadwalan dan Reservasi

Sebelumnya, reservasi pasien dilakukan secara manual melalui WhatsApp, yang dapat menyebabkan miskomunikasi atau data duplikat. Sistem berbasis web ini mengotomatisasi proses reservasi, memastikan bahwa jadwal dokter selalu terupdate dan tidak terjadi kesalahan dalam penjadwalan. Pasien dapat langsung melihat ketersediaan dokter dan memilih waktu yang sesuai serta membuat sistem lebih transparan.

### 4. Fitur Visualisasi Data

Sistem sebelumnya tidak memiliki sistem untuk memantau performa klinik secara terukur, sehingga dapat menghambat *owner* untuk melakukan pemantauan berbasis data terkait produktivitas dokter, tren layanan perawatan, dan tren pasien. Fitur visualisasi data dalam sistem berbasis web ini memungkinkan klinik menghasilkan visualisasi yang dapat diakses kapan saja untuk memantau perkembangan klinik secara berkala. Data yang dihasilkan dapat digunakan untuk mendukung keputusan strategis *owner*.

### 5. Pengurangan Penggunaan Kertas dan Penyimpanan Fisik

Pada sistem sebelumnya, pengelolaan dokumen secara fisik memerlukan banyak ruang penyimpanan dan kebutuhan akan kertas. Selain itu, risiko kerusakan atau kehilangan dokumen juga tinggi. Sistem ini mengurangi kebutuhan akan penyimpanan fisik, memungkinkan klinik menyimpan data secara digital dan aman, sekaligus mendukung keberlanjutan lingkungan dengan mengurangi penggunaan kertas.

Selain itu, pada sistem ini juga terdapat keunggulan atau kebaruan yang dapat memberikan nilai tambah pada tugas akhir ini. Keunggulan dan kebaruan dari sistem ini dapat dilihat sebagai berikut:

#### 1. *PWA untuk Akses Offline*

Sistem ini menggunakan *PWA* yang memungkinkan halaman web tetap diakses meskipun koneksi internet tidak tersedia. Fitur ini meningkatkan aksesibilitas dan memberikan pengalaman seperti aplikasi mobile. Implementasi ini

menggunakan *service workers* untuk mengontrol *caching* dan sinkronisasi data, memungkinkan aplikasi tetap berfungsi secara offline. Pada fitur ini website awalnya akan mencoba mengakses tampilan website melalui server terlebih dahulu, jika website terkoneksi maka akan tampilan akan ditampilkan berdasarkan server. Ketika website tidak terhubung ke server maka website akan menampilkan tampilan dari *cache* yang sudah disimpan sebelumnya, jika cache juga tidak ada maka website akan menampilkan tampilan *default* website offline yang sudah deprogram sebelumnya. Dengan adanya fitur ini maka pengguna dapat mengakses informasi penting tanpa bergantung pada koneksi internet yang stabil.

## 2. Notifikasi WhatsApp (API Twilio)

Sistem ini mengintegrasikan API Twilio untuk memfasilitasi pengiriman notifikasi langsung melalui platform WhatsApp. Integrasi ini bertujuan untuk membantu komunikasi antara sistem dan pengguna. Dengan fitur ini maka klinik bisa mengirim pesan whatsapp secara terprogram. Penggunaan API Twilio juga bisa mengirimkan *template* pesan yang dipersonalisasi sesuai kebutuhan dengan berbagai jenis konten termasuk teks, gambar, video dan *file* serta sistem dapat menangani pengiriman pesan dalam jumlah besar. Dengan adanya fitur ini pasien mendapatkan pemberitahuan terkait jadwal dan reservasi secara real-time, meningkatkan komunikasi antara klinik dan pasien.

## 3. Visualisasi Data dengan Filter

Fitur visualisasi data memudahkan pemilik dan manajemen klinik untuk melihat statistik klinik, seperti statistic pasien, pendapatan bulanan, jumlah kunjungan, kinerja dokter. Data ini dapat difilter berdasarkan bulan untuk analisis kinerja yang lebih detail.



#### 4. Laravel Versi 11

Sistem dibangun menggunakan Laravel 11, yang memberikan peningkatan dalam hal kesederhanaan struktur aplikasi, kinerja, keamanan, kompatibilitas dengan PHP terbaru, kemudahan pengujian, dan konfigurasi.

Disamping keunggulan tersebut, juga terdapat kekurangan pada sistem informasi manajemen klinik yang sudah dibangun ini dan dapat dilakukan pengembangan lebih lanjut menjadi lebih baik. Kekurangan dalam pembangunan sistem ini terletak pada belum tersedianya fitur payment gateway yang memungkinkan pasien melakukan transaksi secara online. Hal ini menjadi penting untuk membantu proses pembayaran dan memberikan kenyamanan lebih bagi pengguna. Selain itu, meskipun sistem telah menggunakan teknologi *PWA*, penerapannya masih terbatas hanya pada fitur offline mode dan diperlukan peningkatan keamanan untuk fitur offline ini. Fitur penting lainnya dari *PWA* seperti *push notification*, *background sync*, dan peningkatan *user experience* lainnya juga belum diimplementasikan.

## **BAB VI**

### **KESIMPULAN DAN SARAN**

Bab ini menguraikan kesimpulan dan saran dari laporan tugas akhir ini. Kesimpulan adalah hasil yang dicapai dari penelitian berdasarkan tujuan yang telah ditetapkan di awal penelitian. Saran adalah rekomendasi yang diharapkan terkait penelitian yang telah dilakukan untuk masa mendatang.

#### **6.1 Kesimpulan**

*Website* sistem informasi manajemen klinik gigi pada klinik Xenon Dental House telah berhasil dibangun menggunakan metode waterfall. Dari penelitian ini, sistem yang dibangun menawarkan solusi dalam mengatasi kendala serta membantu pengelolaan manajemen klinik. Sistem ini mencakup pengelolaan data pasien, dokter, data rekam medis, reservasi, izin dokter, layanan perawatan, dan visualisasi data klinik.

Dengan implementasi sistem ini, proses reservasi yang awalnya dilakukan secara manual menjadi reservasi yang lebih tersistem dan lebih akurat melalui sistem reservasi berbasis website, sehingga meminimalkan risiko miskomunikasi dan kesalahan penjadwalan. Data klinik kini terintegrasi dalam satu pusat sistem tanpa perlu bergantung pada berbagai media pencatatan. Rekam medis digital menggantikan media kertas, mengurangi risiko kerusakan atau kehilangan data, serta menghemat ruang penyimpanan fisik. Penerapan fitur *Progressive Web App* (PWA) memungkinkan akses *offline*, menjadikan sistem tetap bisa diakses meskipun tidak terhubung ke server. Selain itu, integrasi *API* Twilio mendukung pengiriman notifikasi untuk mengingatkan serta memberikan informasi penting untuk user melalui WhatsApp.

Sistem ini tidak hanya membantu pengelolaan klinik secara operasional tetapi juga membuka peluang untuk membantu pengambilan keputusan melalui visualisasi data *real-time* yang disediakan *website*. Dengan demikian, implementasi sistem informasi manajemen ini berhasil dibangun untuk mengatasi kendala dalam manajemen klinik di Klinik Gigi Xenon Dental House dan memberikan solusi untuk mendukung pengelolaan data.

## 6.2 Saran

Sistem informasi manajemen klinik gigi pada klinik Xenon Dental House ini masih membutuhkan pengembangan lebih lanjut pada fungsional aplikasi. Saat ini *website* yang dibangun hanya fokus pada penerapan *website* pada bisnis proses klinik gigi xenon dental house. Saran terhadap pembangunan *website* ini selanjutnya adalah pembangunan fitur payment gateway pada *website* ini agar bisa dilakukan transaksi secara online. Pembangunan lebih lanjut juga dengan menerapkan fitur *PWA* lainnya seperti *push notification*, *background sync* dan fitur lainnya yang ada pada *PWA* karena pada pembangunan *website* ini hanya berfokus pada fitur akses offline *PWA* saja serta meningkatkan keamanan website untuk fitur *PWA offline*.

## DAFTAR PUSTAKA

- 'Afifah, K., Azzahra, Z.F. and Anggoro, A.D. (2022) 'Analisis Teknik Entity-Relationship Diagram dalam Perancangan Database Sebuah Literature Review' doi:10.54895/intech.v3i2.1682.
- Aleksandrs Hodakovskis; Katrina Antonova (2019) 'The ultimate guide to Progressive Web Apps', Scandiweb.
- Amalia, R. and Huda, N. (2020) 'Implementasi Sistem Informasi Pelayanan Kesehatan Pada Klinik Smart Medica', Jurnal Sisfokom (Sistem Informasi dan Komputer). doi:10.32736/sisfokom.v9i3.884.
- Astuti, R. (2009) 'Pemodelan Analisis Berorientasi Objek dengan Use Case', Media Informatika. Available at: [https://jurnal.likmi.ac.id/Jurnal/7\\_2009/Pemodelan\\_Analisis\\_rini\\_.pdf](https://jurnal.likmi.ac.id/Jurnal/7_2009/Pemodelan_Analisis_rini_.pdf).
- Cahyani, M.B. et al. 2024. 'Tinjauan Literatur : Peran Rekam Medis Berbasis Elektronik Terhadap Pelayanan Kesehatan'. doi:10.33560/jmiki.v12i2.648.
- Dharwiyanti, S. (2003) 'Pengantar Unified Modeling Language (UML)'
- Hariyanto, S. (2018) 'Sistem Informasi Manajemen', Sistem Informasi Manajemen. Available at: <https://jurnal-unita.org/index.php/publiciana/article/viewFile/75/69>.
- Haryanto, D. and Saputra Elsi, Z.R. (2021) 'Analisis Performance Progressive Web Apps Pada Aplikasi Shopee', Jurnal Ilmiah Informatika Global. /doi:10.36982/jiig.v12i2.1944.
- Indah, P.N. et al. (2021) 'PERAN TEKNOLOGI INFORMASI PADA PERUBAHAN ORGANISASI DAN FUNGSI AKUNTANSI MANAJEMEN', JRAK (Jurnal Riset Akuntansi dan Bisnis). doi:10.38204/jrak.v7i2.625.
- Isana, B., Kumboyono, K. and Windarwati, H.D. (2022) 'Pengalaman Perawat Bekerja dengan Sistem Informasi Kesehatan dan EHR : Scoping Review Nurse Experience Working with Health Information Systems and EHR :

Scoping Review’, Jurnal Kesehatan Vokasional. Available at:  
<https://journal.ugm.ac.id/jkesvo/article/view/73874>.

Ismanto, I., Hidayah, F. and Charisma, K. (2020) ‘Pemodelan Proses Bisnis Menggunakan Business Process Modelling Notation (BPMN) (Studi Kasus Unit Penelitian Dan Pengabdian Kepada Masyarakat (P2KM) Akademi Komunitas Negeri Putra Sang Fajar Blitar)’, Briliant: Jurnal Riset dan Konseptual. doi:10.28926/briliant.v5i1.430.

Karli, T., Muawwal, A. and Thayf, M.S.S. (2023) ‘Implementasi Progressive Web Application Pada Website Frizfoo Menggunakan Express.js’, KHARISMA Tech, 18(2). doi:10.55645/kharismatech.v18i2.445.

Kemenkes RI 2011 (2011) ‘Permenkes No. 028 tentang Klinik 2011’, Menteri Kesehatan Republik Indonesia Peraturan Menteri Kesehatan Republik Indonesia, Nomor 65(879).

Khoirunnisa, I., Pangestu, A.D. and Saputra, E. (2021) ‘Perancangan Sistem Administrasi dan Catatan Rekam Medik Pasien pada Klinik Putri Husada’, Jurnal Riset dan Aplikasi Mahasiswa Informatika (JRAMI). doi:10.30998/jrami.v2i01.1090.

Kurniawan Ritonga, R. and Firdaus, R. (2024) ‘Pentingnya Sistem Informasi Manajemen Dalam Era Digital the Importance of Management Information Systems in the Digital Era’, JICN: Jurnal Intelek dan Cendekiawan Nusantara. Available at: <https://jicnusantara.com/index.php/jicn>.

Lestari, D. (2019) ‘Analisa Dan Perancangan Aplikasi Sistem Pelayanan Klinik Gigi (Studi Kasus: Dental Echo Clinic)’, JSAI (Journal Scientific and Applied Informatics). doi:10.36085/jsai.v2i1.158.

Majidah, R., Rusdianto, D.S. (2019) ‘Pengembangan Sistem Informasi Pengelolaan Klinik Gigi Berbasis Website Menggunakan Prinsip Point of Sale

Media Informatika.2009. Pemodelan Analisis Berorientasi Objek Dengan.  
Available at:

[https://jurnal.likmi.ac.id/Jurnal/7\\_2009/Pemodelan\\_Analisis\\_rini\\_.pdf](https://jurnal.likmi.ac.id/Jurnal/7_2009/Pemodelan_Analisis_rini_.pdf).

Muddin, S., Tehuayo, H. and Iksan, F. (2021) ‘Penerapan Teknologi Progressive Web Apps (PWA) Pada Sistem Informasi Sma Negeri 7 Buru Selatan’, Jurnal Teknologi dan Komputer (JTEK). doi:10.56923/jtek.v1i01.48.

Muntihana, V. et al. (2017) Berbasis Web Dan Android Pada Klinik Gigi Lisda.

Novianti Indah. 2021. Peran Teknologi Informasi Pada Perubahan Organisasi Dan Fungsi Akuntansi Manajemen. 1

Nuryani, S. (2021) ‘Pengembangan Aplikasi Mobile Booking Online Perawatan Gigi Dengan Metode Prototype Studi Kasus Di Klinik Gigi Budiono, Drg. Kota Bandung’, Jurnal Ekonomi, Sosial & Humaniora.

Pricillia, T. and Zulfachmi (2021) ‘Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD)’, Jurnal Bangkit Indonesia. doi:10.52771/bangkitindonesia.v10i1.153.

Pulungan, S.M. et al. (2023) ‘Analisis Teknik Entity-Relationship Diagram Dalam Perancangan Database’, Jurnal Ekonomi Manajemen dan Bisnis (JEMB). doi:10.47233/jemb.v1i2.533.

Rendra Wisiasto 2010. Perancangan Dan Implementasi Sistem Penjadwalan Konsultasi Dokter Terhadap Pasien Via Sms Gateway Di Rumah Sakit Rajawali. Universitas Komputer Indonesia.

Riandy, G., Edi, S., & Rengga, A. (2009). Rancang Bangun Sistem Informasi Reservasi Online Fisioterapi Menggunakan JSP Sebagai Sistem Pelayanan Terpadu. EEPIS Final Project.

Saptiono Mulana, V.A. (2020) ‘Identifikasi Proses Bisnis di Instalasi Rawat Inap Rumah Sakit Universitas Gadjah Mada Dengan Pendekatan Grounded Theory’, Journal of Information Systems for Public Health. doi:10.22146/jisph.7603.

Sari, Ira Puspita & Diki Arisandi.(2017). Sistem Informasi Manajemen Klinik Gigi Berbasis Client Server.

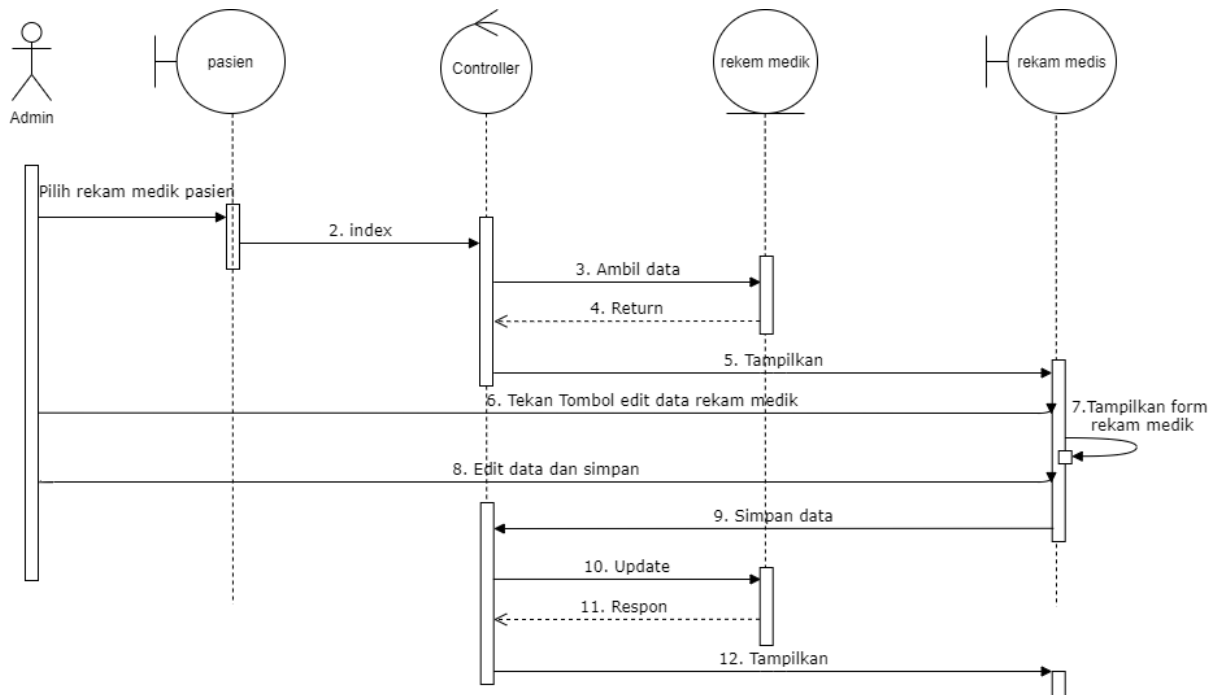
- Setyadi, H.A. and Perbawa, D.S. (2021) ‘Sistem Informasi Rekam Medis Di Klinik Gigi Rumah Sakit Paru dr. Ario Wirawan Salatiga’, Jurnal Infortech. doi:10.31294/infortech.v3i2.11209.
- Silberschatz, A., Korth, H.F. and Sudarshan, S., 2011. Database system concepts. 6th ed. New York: McGraw-Hill.
- SOMMERVILLE, I. (2011) Requisitos de ingeniería de software.
- Stiehl, V., Raw, R. and Smith, P. (2014) Process-driven applications with BPMN, Process-Driven Applications with BPMN. doi:10.1007/978-3-319-07218-0.
- Tambunan, J.C. and Malau, E.P. (2022) ‘Sistem Informasi Klinik Berbasis Web’, KAKIFIKOM (Kumpulan Artikel Karya Ilmiah Fakultas Ilmu Komputer). doi:10.54367/kakifikom.v4i1.1873.
- Wahyudi, T., Supriyanta, S. and Faqih, H. (2021) ‘Pengembangan Sistem Informasi Presensi Menggunakan Metode Waterfall’, Indonesian Journal on Software Engineering (IJSE).
- Wagner, G. (2017) ‘Information and Process Modeling for Simulation – Part I: Objects and Events’, Journal of Simulation Engineering. Available at: <https://articles.jsime.org/1/1>.
- Wardianto, M. (2011). Rancang bangun aplikasi pendapatan online jasa pengobatan berbasis multimedia pada Klinik Utama Siti Aksar Depok.
- Warrender, R., Links, U. and Dutton, S. (2018) ‘Progressive Web Apps: the future of the mobile web’.
- Wijoyo, H. (2021) Sistem Informai Manajemen, Buku.
- Yohana, N.D. and Marisa, F. (2018) ‘Perancangan Proses Bisnis Sistem Human Resource Management (HRM) Untuk Meningkatkan Kinerja Pegawai’, J I M P - Jurnal Informatika Merdeka Pasuruan. doi:10.37438/jimp.v3i2.168.
- Yossiant, S. and Hosizah, H. (2023) ‘Implementasi Rekam Medis Elektronik di Klinik Kidz Dental Care’, Indonesian of Health Information Management Journal (INOHIM). doi:10.47007/inohim.v11i1.498.

**LAMPIRAN A**  
***SEQUENCE DIAGRAM***  
**(LANJUTAN)**

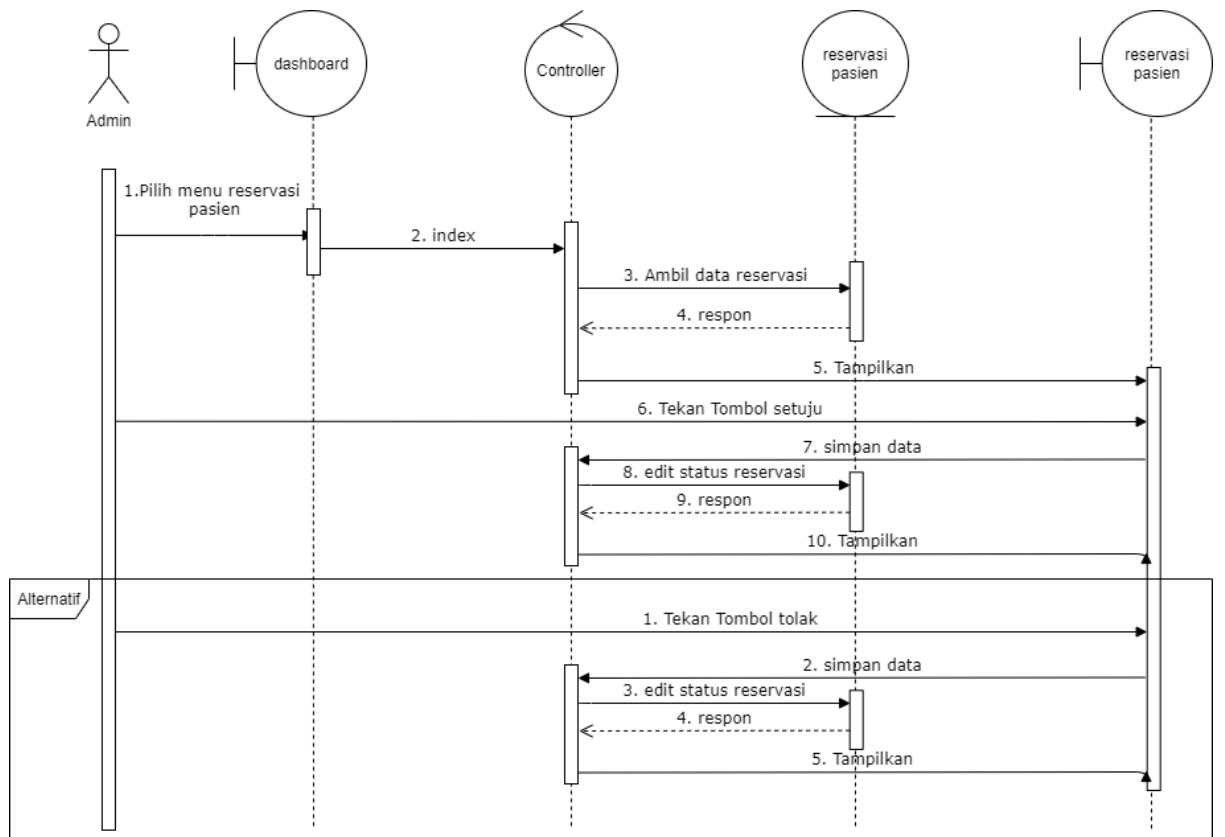


## Admin

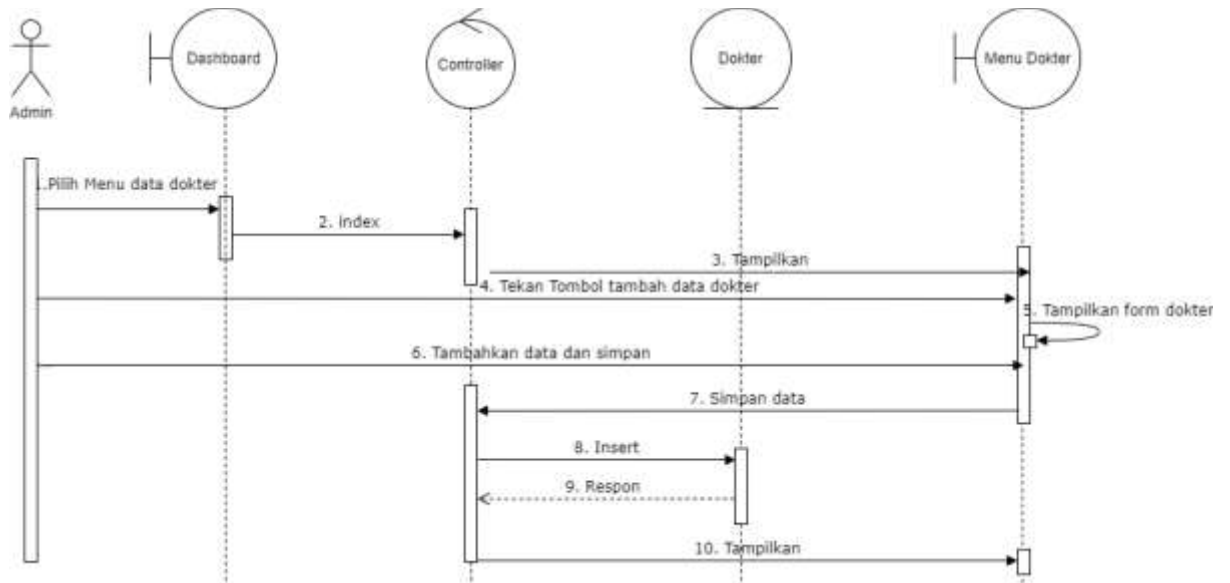
### 1. Edit rekam medik



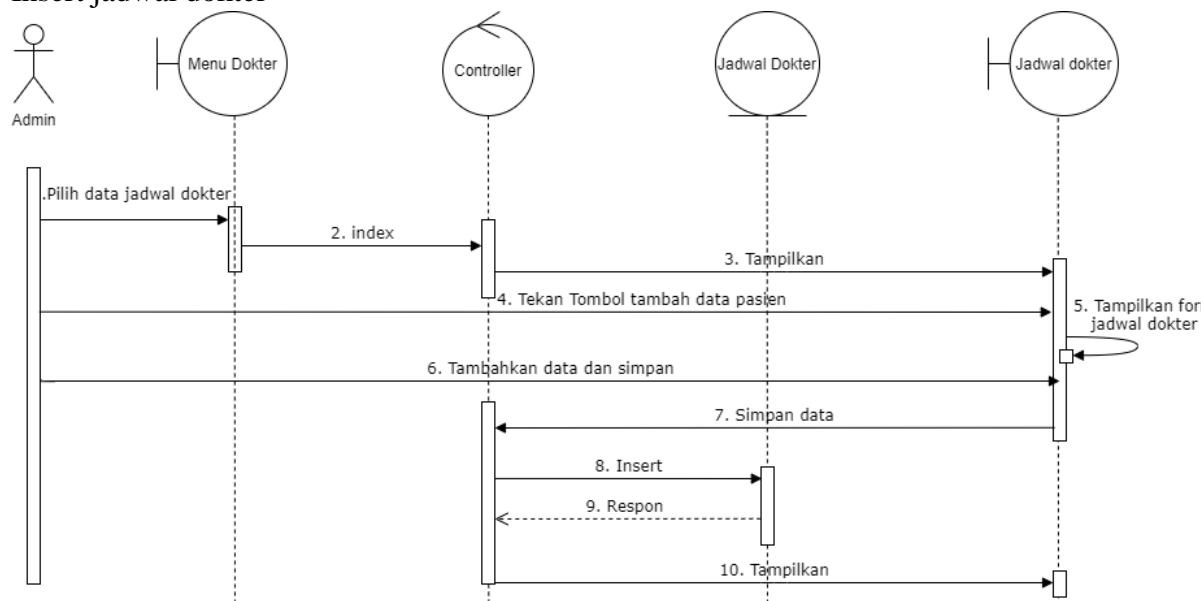
### 2. Menyetujui reservasi



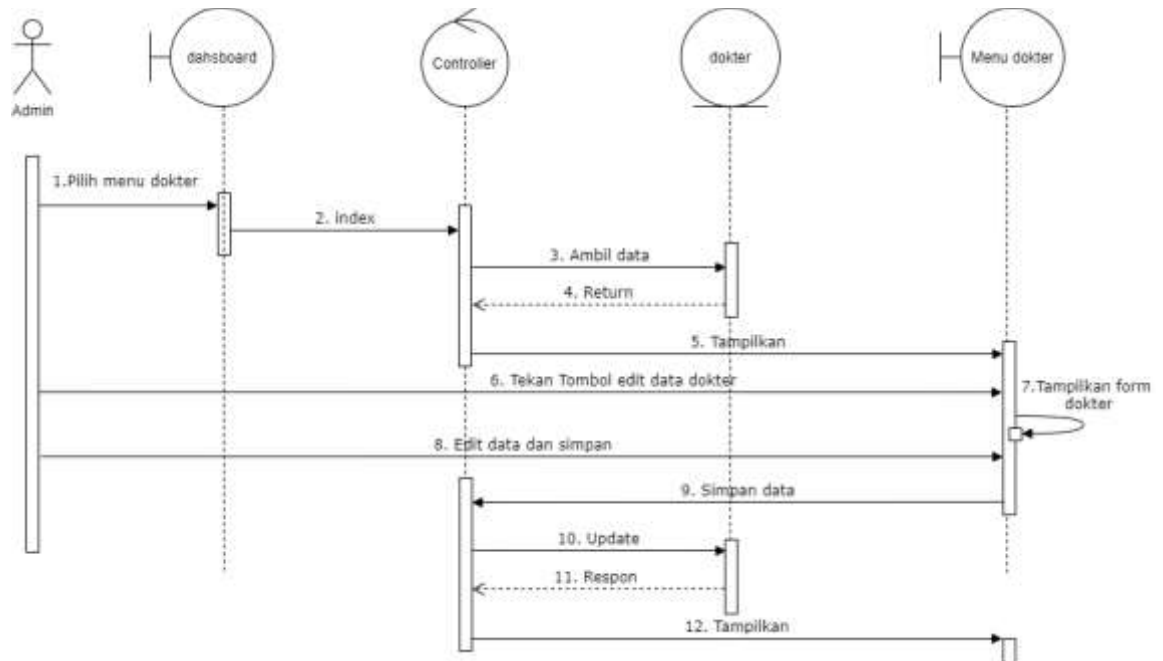
### 3. Insert data dokter



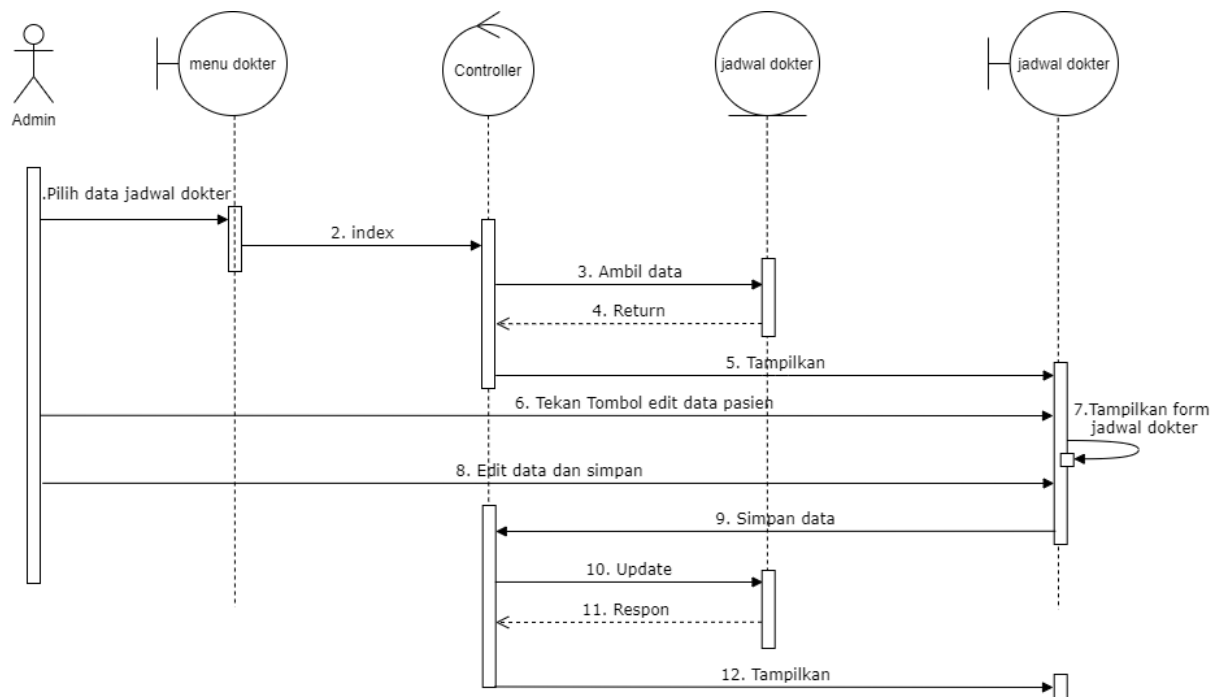
### 4. Insert jadwal dokter



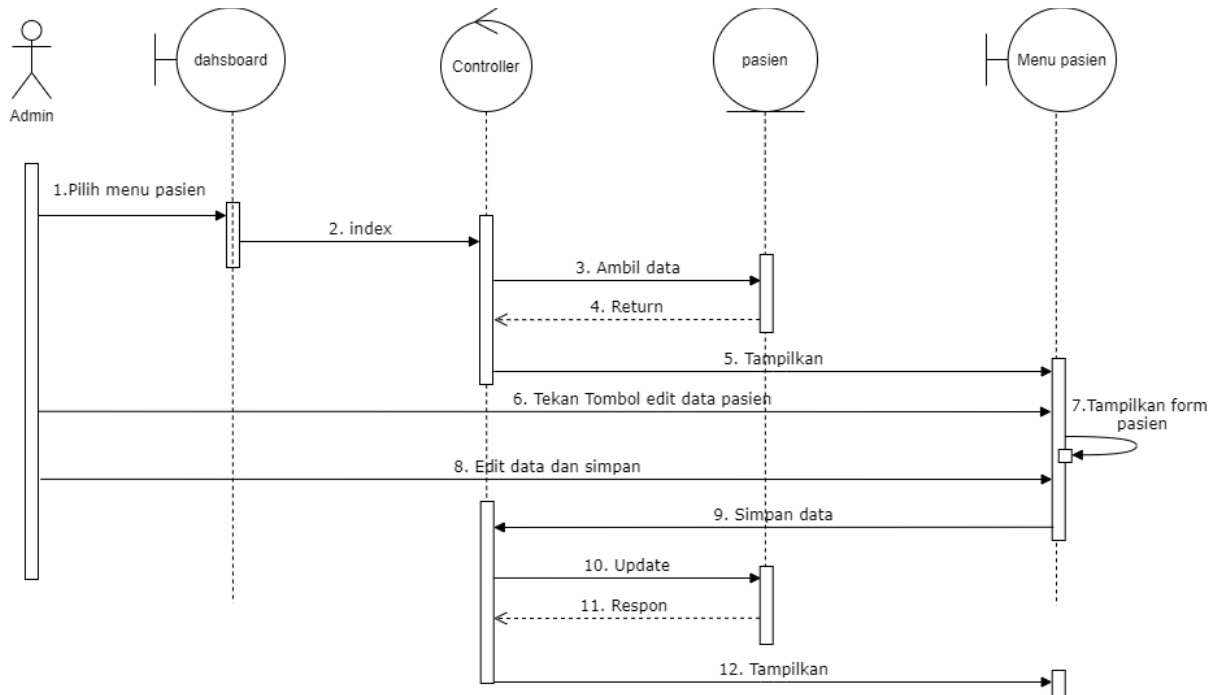
## 5. Edit data dokter



## 6. Edit dokter jadwal

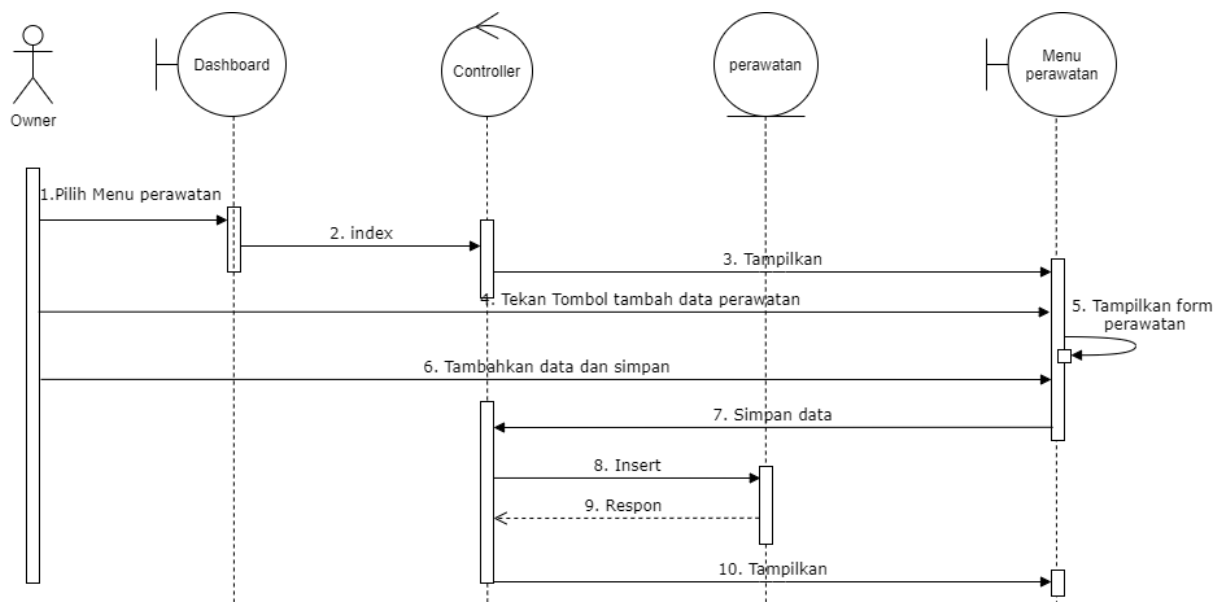


## 7. Update data pasien

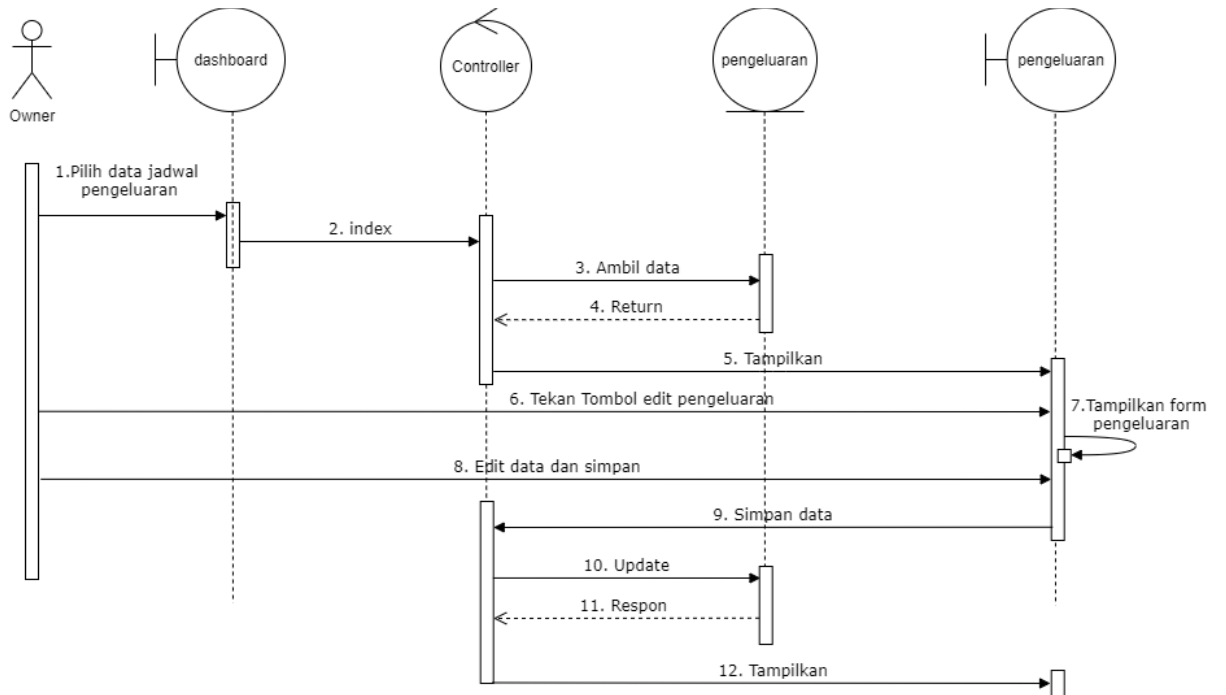


## User Owner

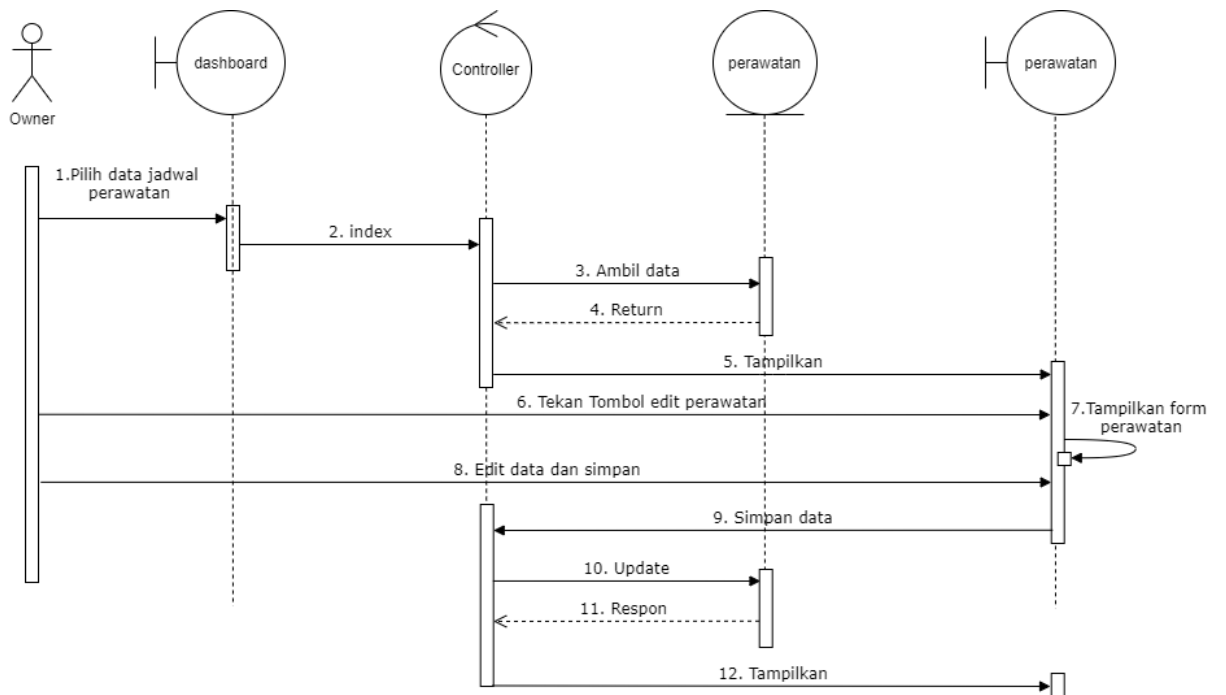
### 8. Insert perawatan



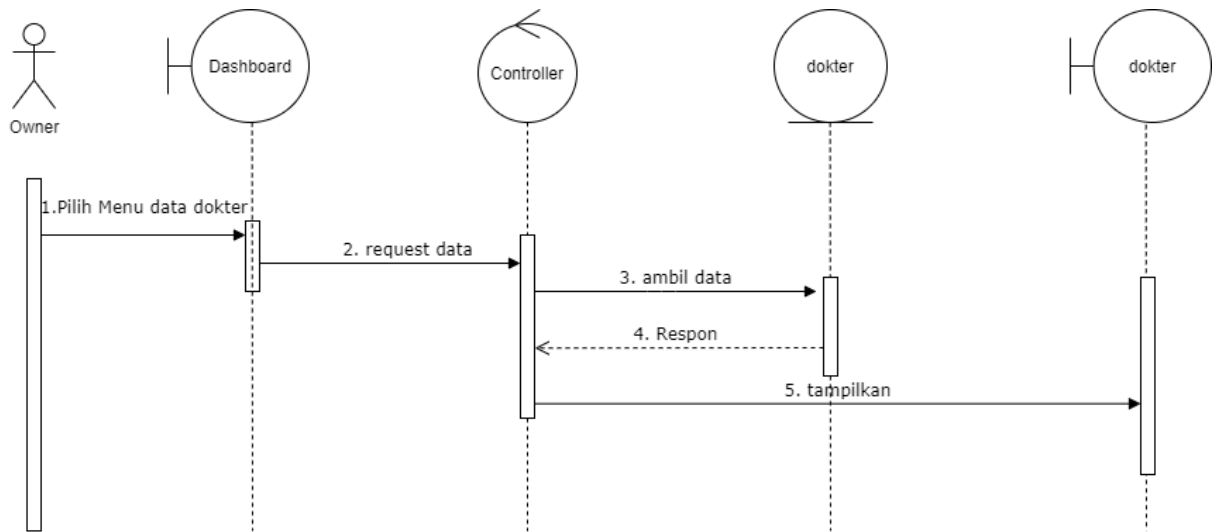
## 9. Edit pengeluaran



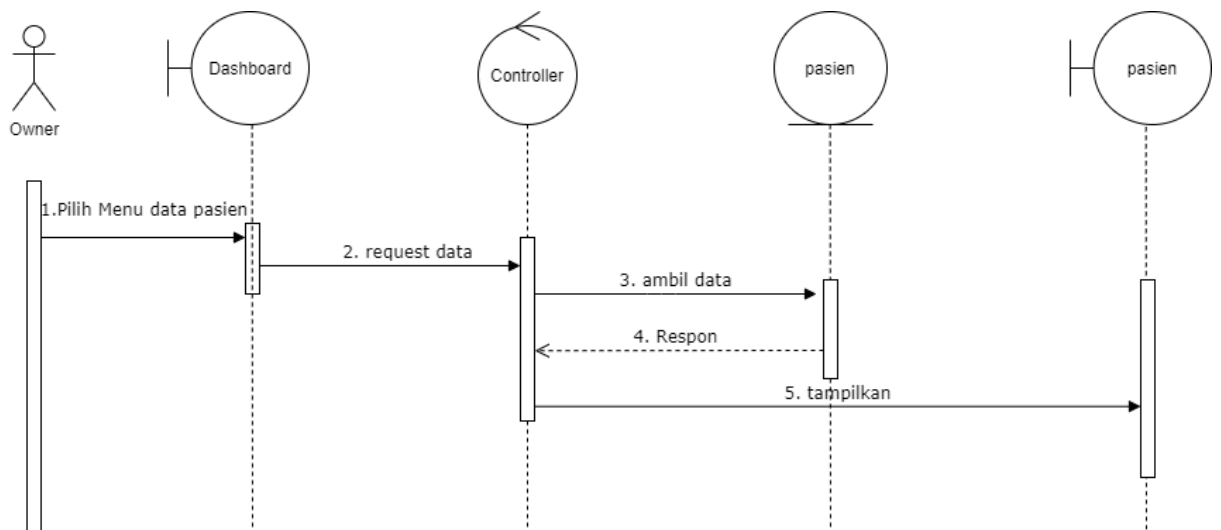
## 10. Edit Perawatan



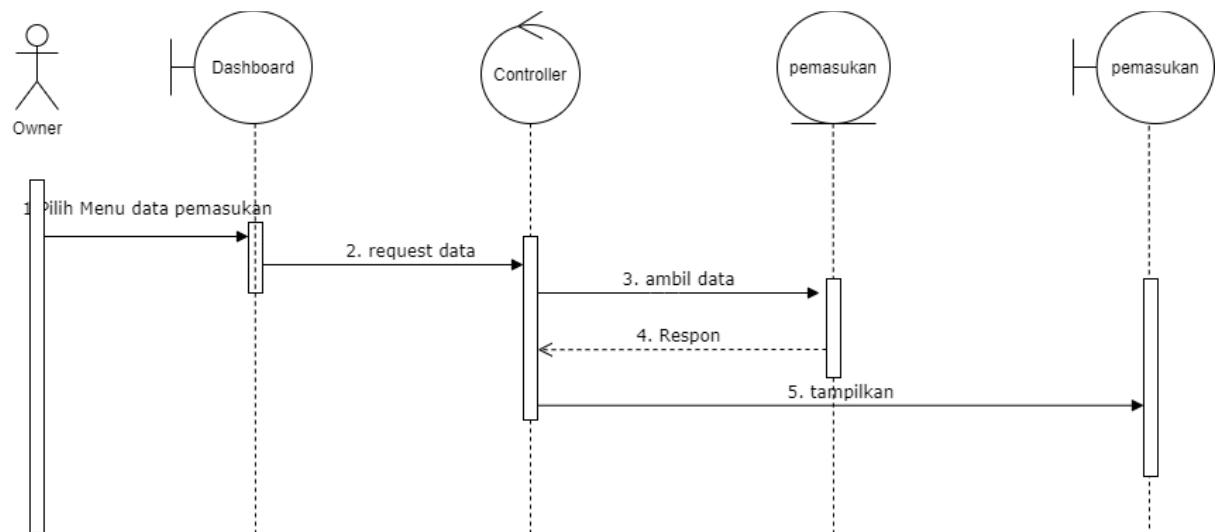
## 11. Menampilkan data dokter



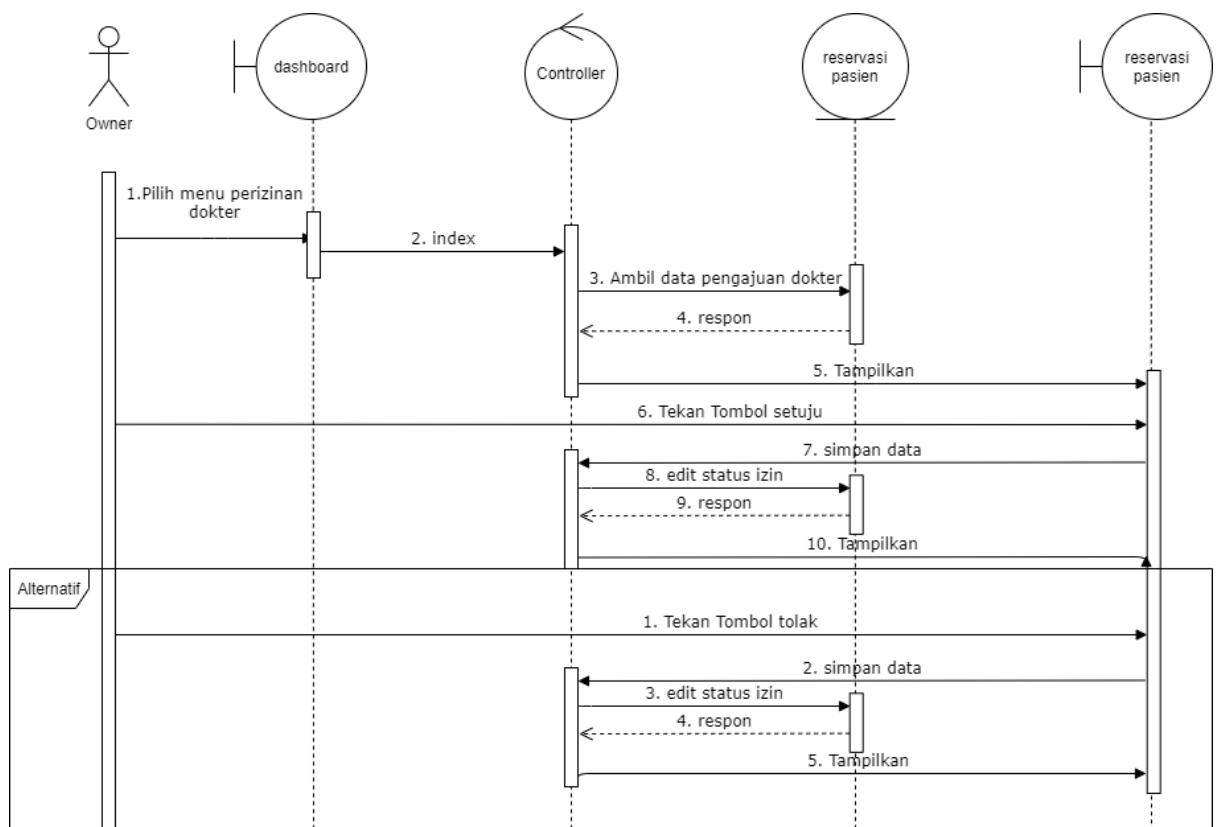
## 12. Menampilkan data pasien



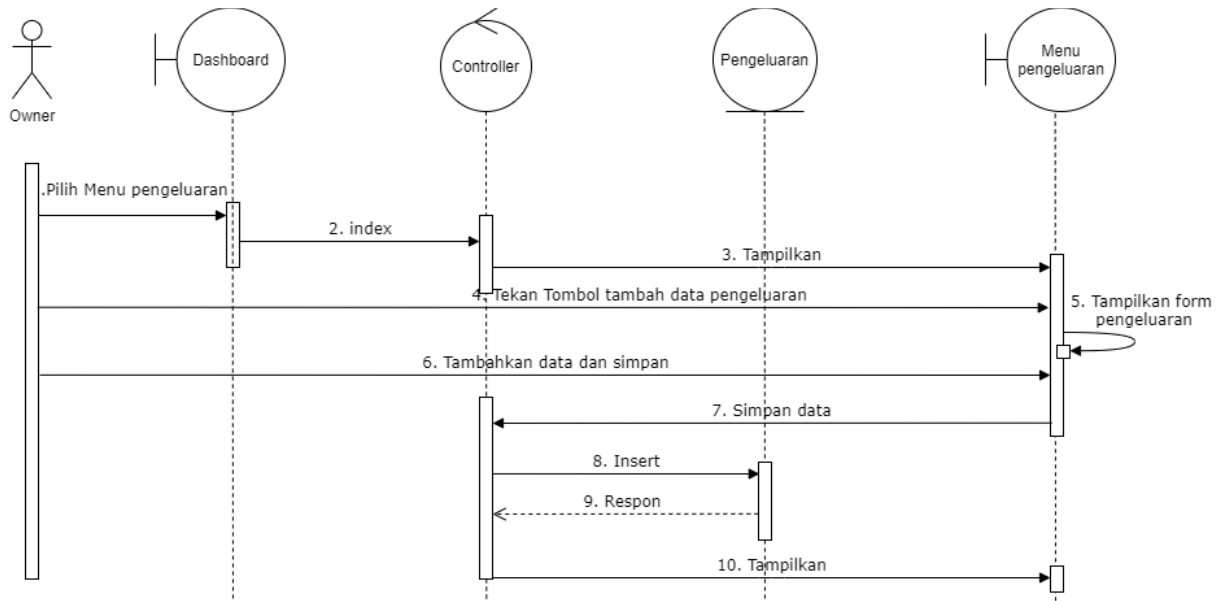
### 13. Menampilkan data pemasukan



### 14. Konfirmasi perizinan oleh owner dari dokter

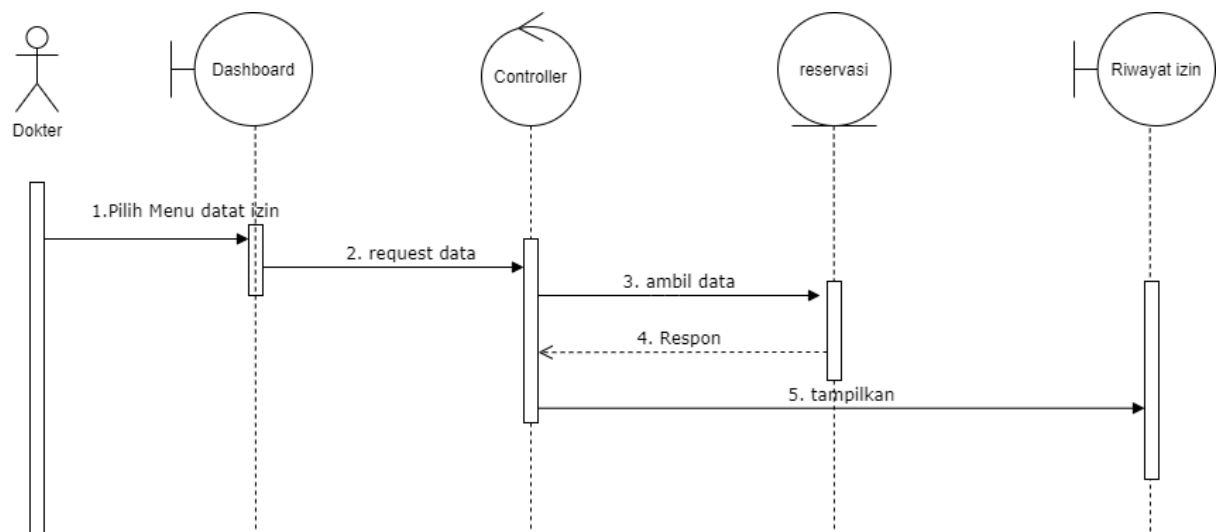


## 15. Menambahkan data pengeluaran



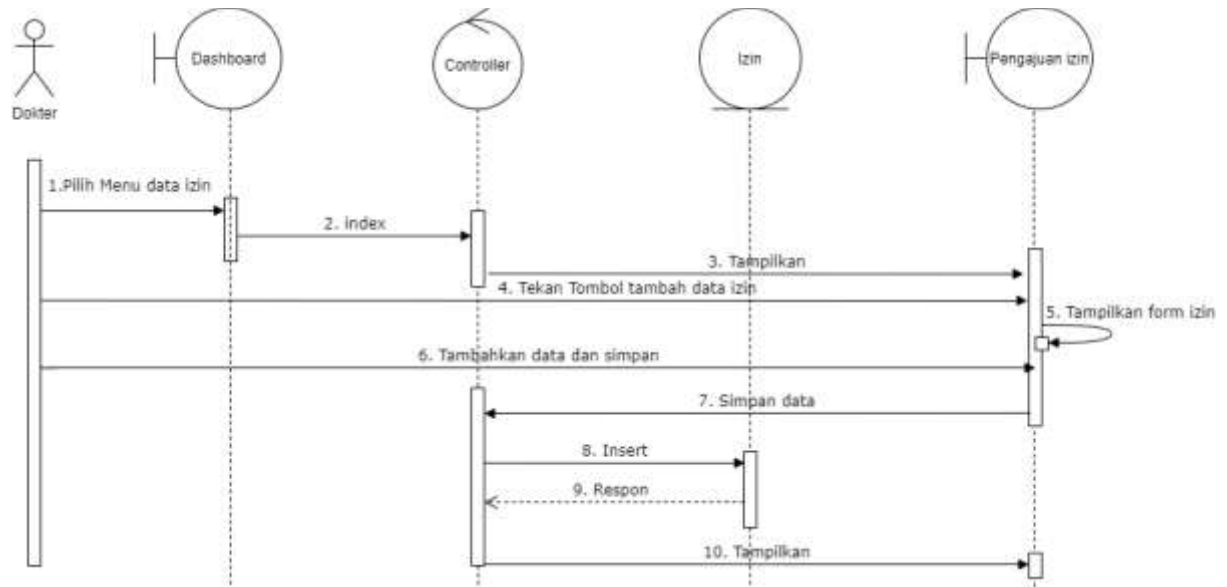
dokter

## 16. Menampilkan Riwayat izin dokter

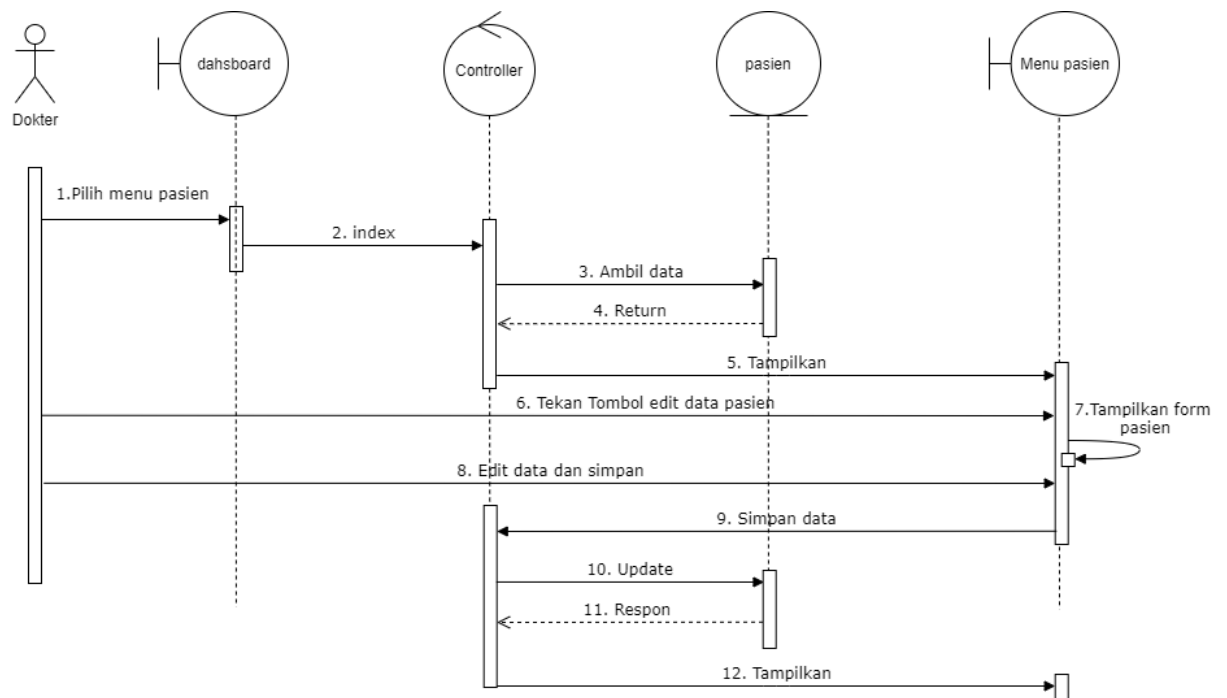




## 17. Pengajuan izin kepada owner

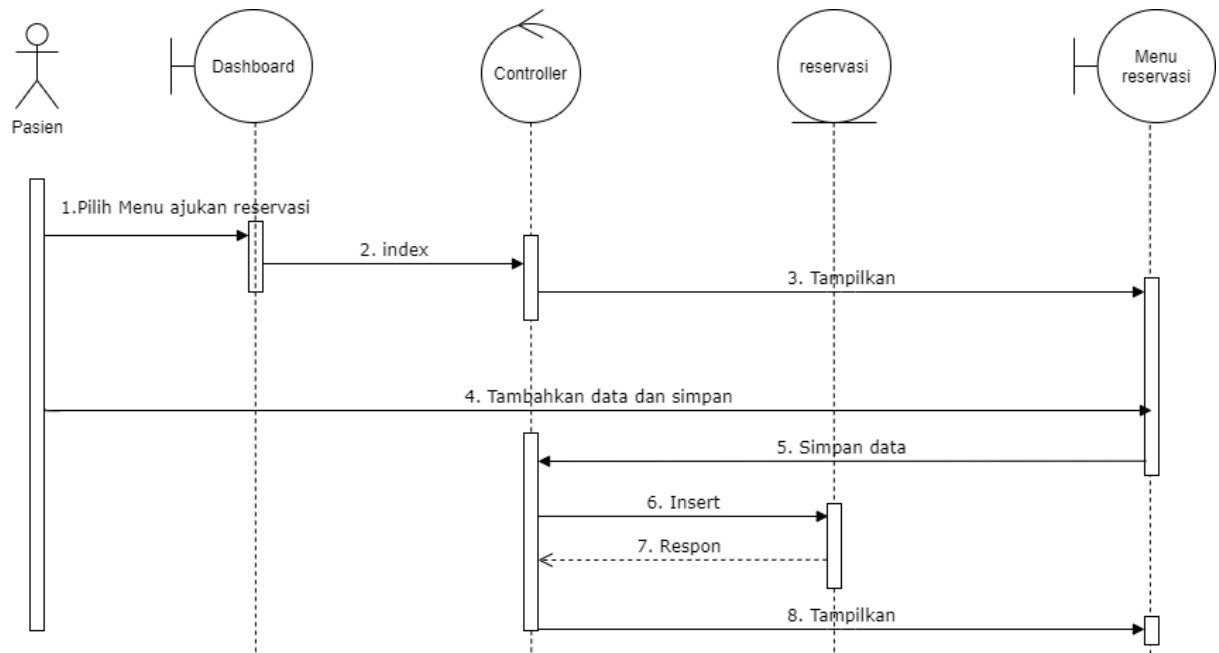


## 18. Edit data pasien

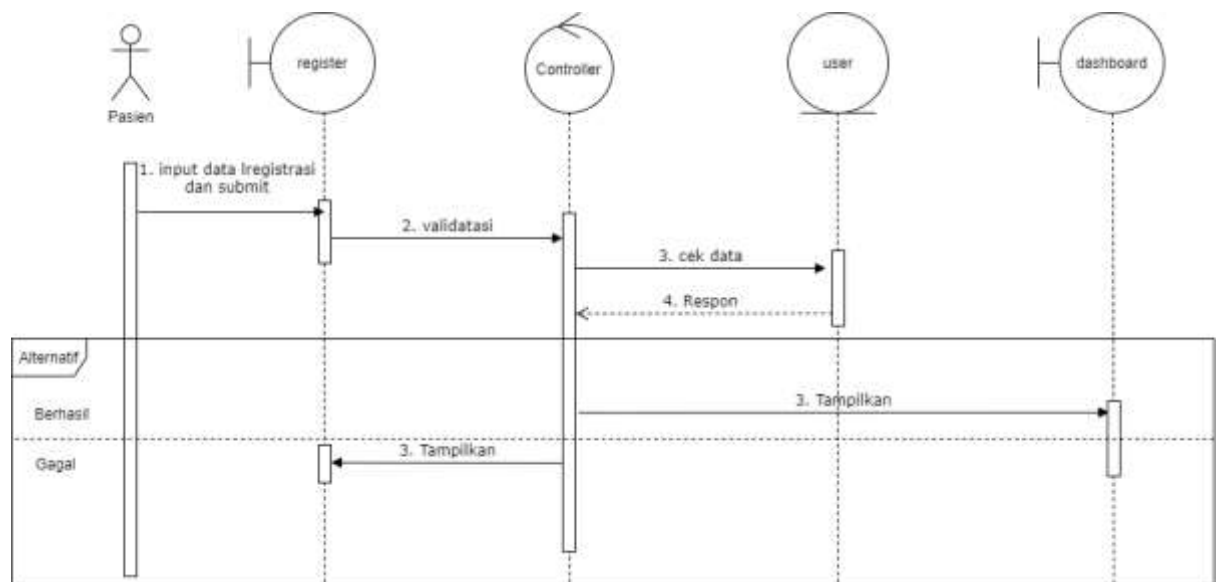


Pasien

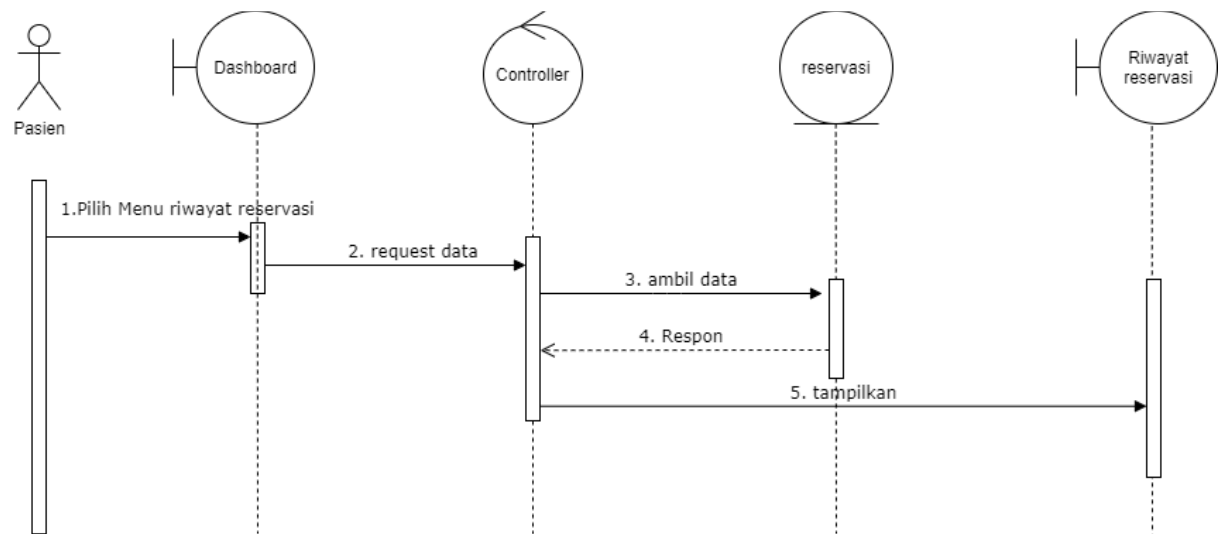
### 19. Menambahkan data reservasi



### 20. Registrasi akun pasien

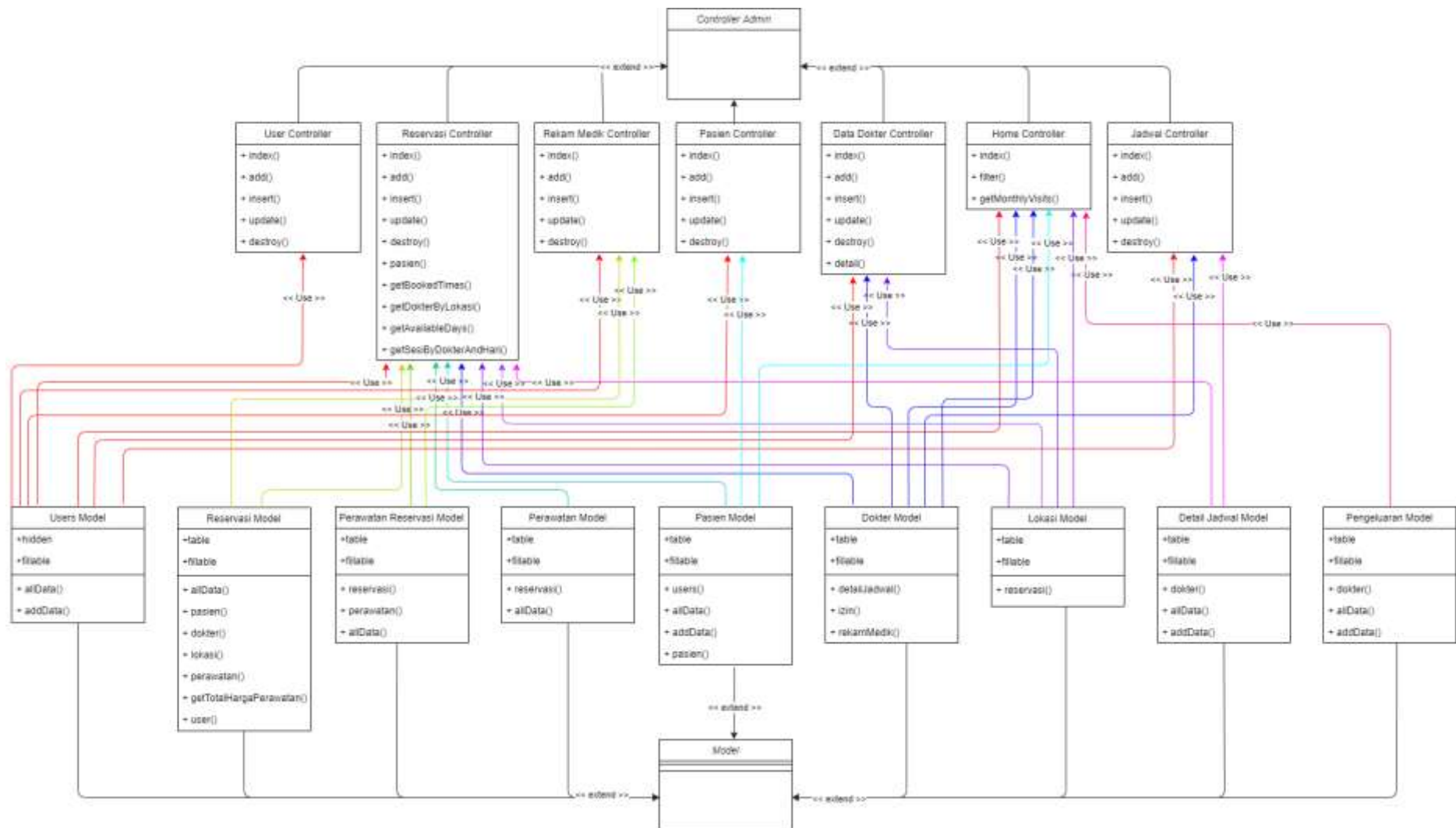


## 21. Menampilkan Riwayat Reservasi

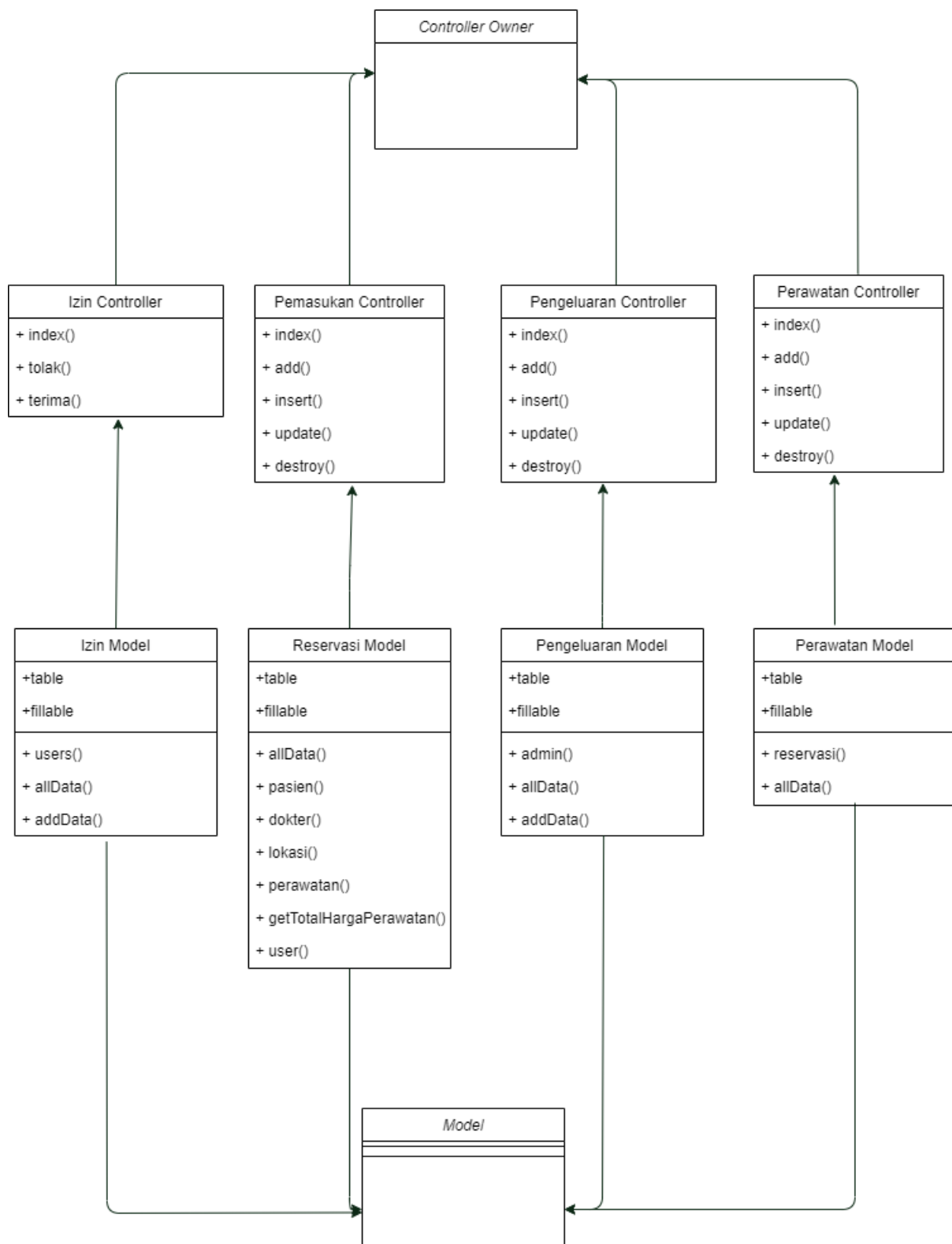


**LAMPIRAN B**  
***CLASS DIAGRAM***  
**(LANJUTAN)**

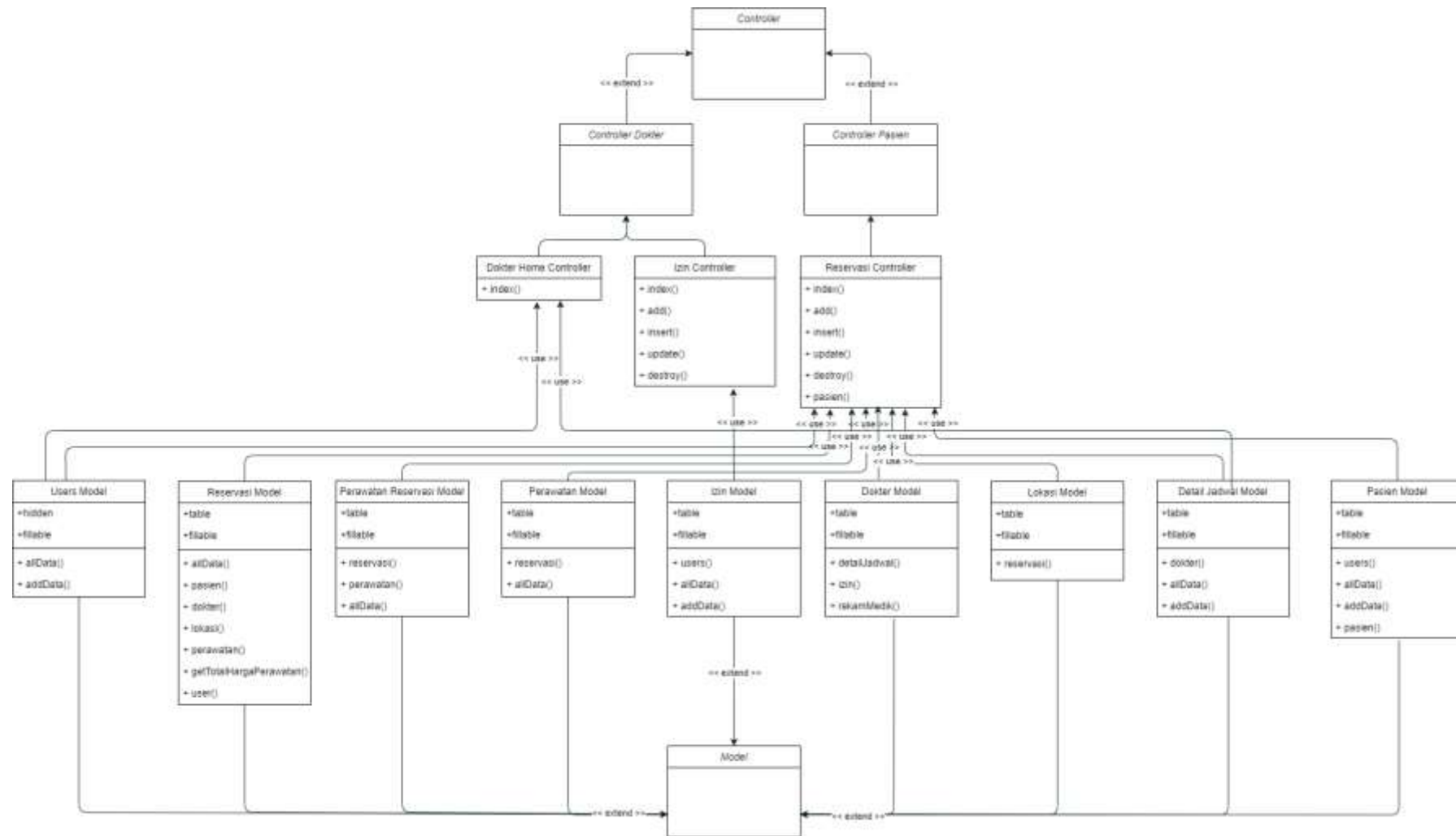
## 1. Class Diagram Admin



## 2. Class Diagram Owner



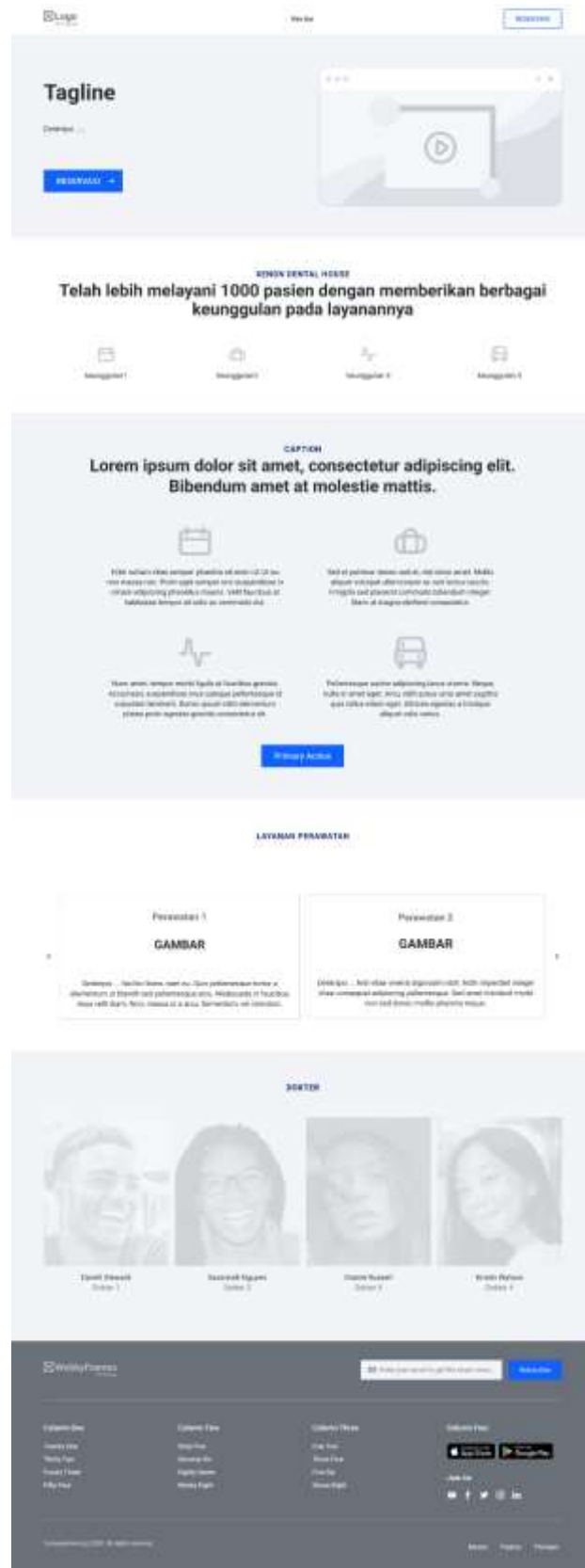
### 3. Class Diagram Pasien dan Dokter



**LAMPIRAN C**  
***MOCKUP* ANTARMUKA**  
**(LANJUTAN)**



## 1. Mockup Antarmuka Halaman Landing Page Setelah Login



## 2. Mockup Antarmuka Halaman registrasi

The mockup shows a 'Sign Up' page with a white background and a light blue border. At the top, the title 'Sign Up' is centered in bold. Below it, there are two input fields for 'First Name' and 'Last Name', each with a 'Placeholder' label. This is followed by an 'Email' field and a 'Password' field with a toggle icon. A blue 'Submit Text' button is positioned below the password field. Underneath the button is a horizontal line, followed by the text 'Or sign up with:'. Below this are three buttons for 'Google', 'Apple', and 'Twitter'. At the bottom, there is a link that says 'Sudah punya akun? login'.

## 3. Mockup Antarmuka Halaman Login

The mockup shows a 'Welcome Back' login page with a white background and a light blue border. At the top, the title 'Welcome Back' is centered in bold, with the subtitle 'Silahkan Log in' below it. There are two input fields for 'Email Address' and 'Password', each with a 'Placeholder' label. The password field has a toggle icon. Below the password field is a small line of text: 'Email dan password harus sesuai dengan yang terdaftar'. A blue 'Log in' button is positioned below this text. Underneath the button is a horizontal line, followed by the text 'Or log in with:'. Below this are three buttons for 'Google', 'Apple', and 'Twitter'. At the bottom, there is a link that says 'No account yet? Sign Up'.

#### 4. Mockup Antarmuka Halaman Profile

**XENON DENTAL HOUSE**

Admin **A**

**My Account**

Name:

Last Name:

Email Address:

Current Password:

New Password:

Confirm Password:

**Save Changes**

**Sidebar:**

- Dashboard
- Reservasi
- Data Pasien
- Data Dokter
- User
- Profile**

#### 5. Mockup Antarmuka Halaman Jadwal Dokter

**XENON DENTAL HOUSE**

Dokter **D**

**Jadwal Saya**

Hari	Sesi
Senin	1
Selasa	2
Rabu	1
Rabu	2
Kamis	1

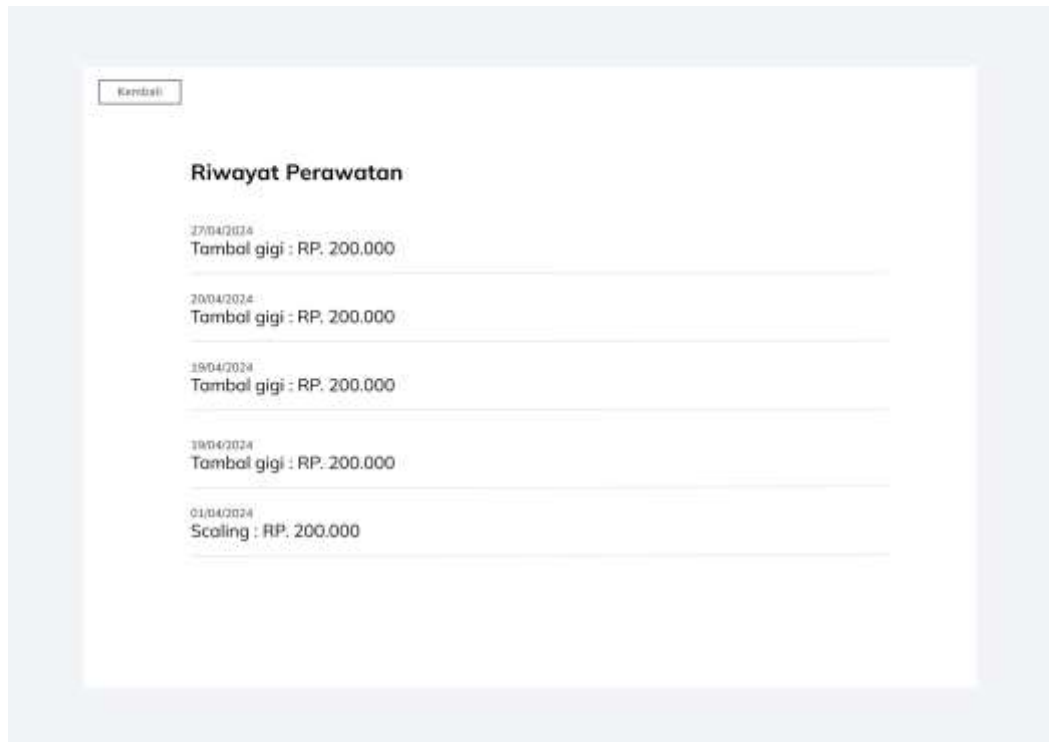
**Shift 1 : 10:00 - 15:00 WIB**

**Shift 2 : 15:00 - 21:00 WIB**

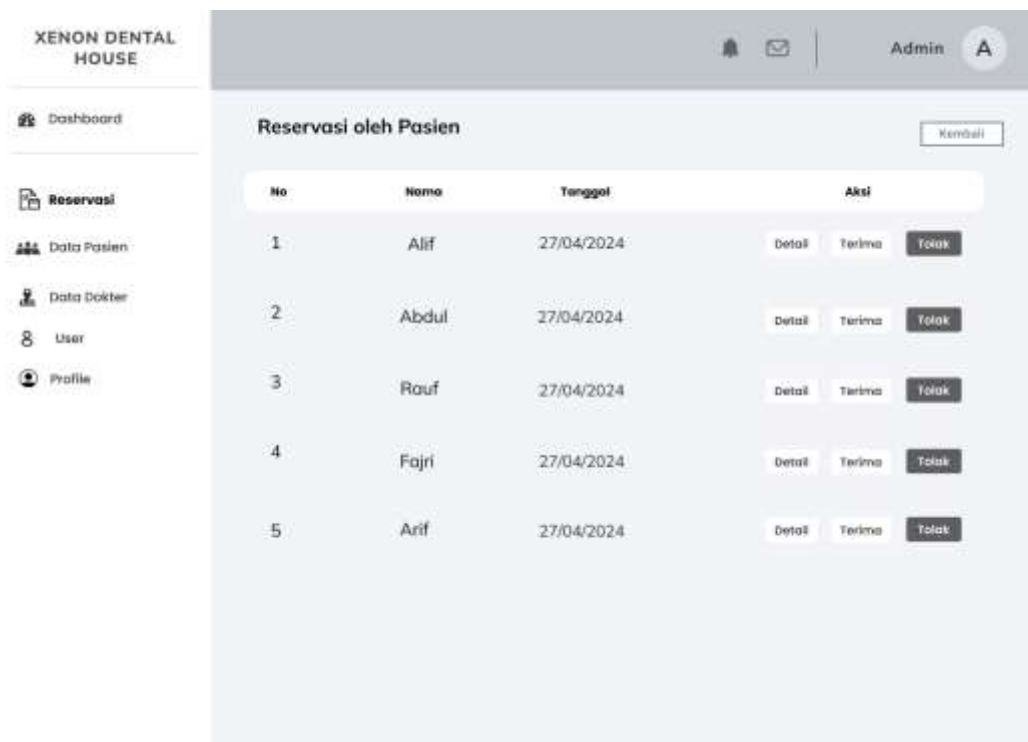
**Sidebar:**

- Jadwal**
- Data Pasien
- Data Absen

## 6. Mockup Antarmuka Halaman Riwayat Perawatan



## 7. Mockup Antarmuka Halaman Reservasi Pasien



## 8. Mockup Antarmuka Halaman Tambah Reservasi

XENON DENTAL  
HOUSE

Dashboard

Reservasi

Data Pasien

Data Dokter

User

Profile

Admin

A

Pengisian Reservasi

Kembali

Tempat

Parkir

Salitings

Nama Dokter

Idcard

Jenis Perawatan

Nomor HP

Email

Submit

## 9. Mockup Antarmuka Halaman Data Pasien User Admin

XENON DENTAL  
HOUSE

Dashboard

Reservasi

Data Pasien

Data Dokter

User

Profile

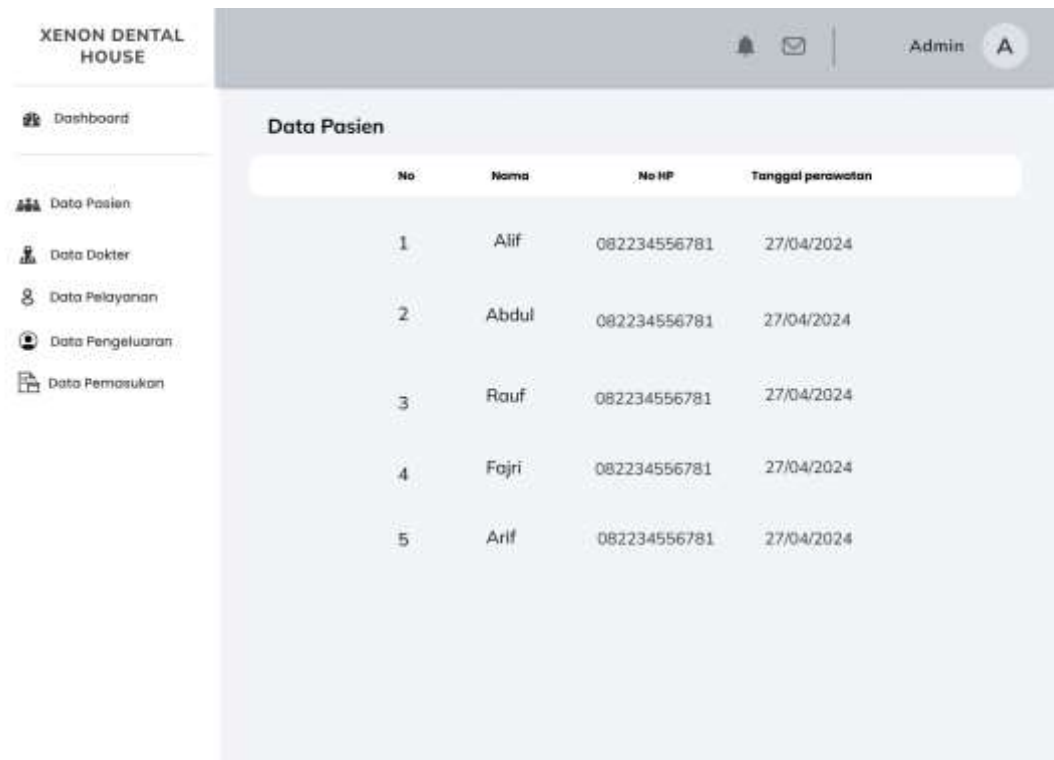
Admin

A

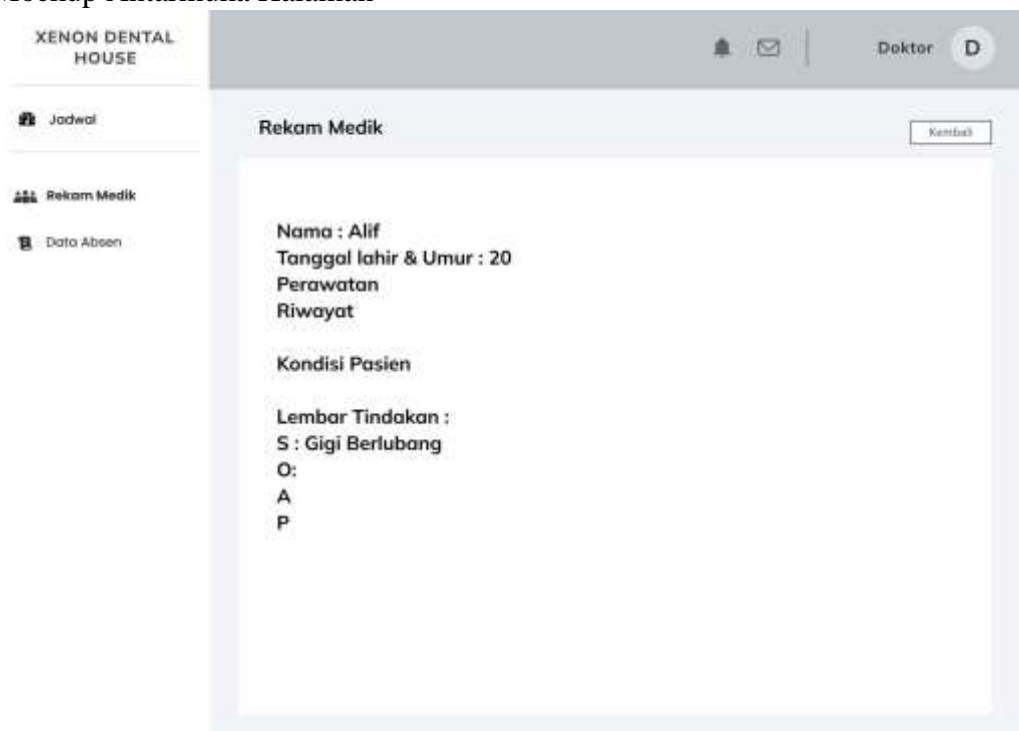
Data Pasien

No	Nama	No HP	Tanggal perawatan	Rekam Medik	Aksi
1	Alif	082234556781	27/04/2024	<div>Q</div>	<div>edit</div> <div>Hapus</div>
2	Abdul	082234556781	27/04/2024	<div>Q</div>	<div>edit</div> <div>Hapus</div>
3	Rauf	082234556781	27/04/2024	<div>Q</div>	<div>edit</div> <div>Hapus</div>
4	Fajri	082234556781	27/04/2024	<div>Q</div>	<div>edit</div> <div>Hapus</div>
5	Arif	082234556781	27/04/2024	<div>Q</div>	<div>edit</div> <div>Hapus</div>

## 10. Mockup Antarmuka Halaman Data Pasien User Owner





## 11. Mockup Antarmuka Halaman






## 12. Mockup Antarmuka Halaman edit Rekam Medik

**XENON DENTAL  
HOUSE**

 Jadwal

 Rekam Medik

 Data Absen

Dokter **D**

### Edit Rekam Medik

Kembali

Nama:

Tanggal lahir / Umur

Riwayat Penyakit / alergi

Dokter

Jenis Perawatan


Tindakan


Biaya Perawatan


Simpan



## 13. Mockup Antarmuka Halaman Izin Dokter

**XENON DENTAL  
HOUSE**

 Jadwal

 Data Pasien

 Data Absen

Dokter **D**

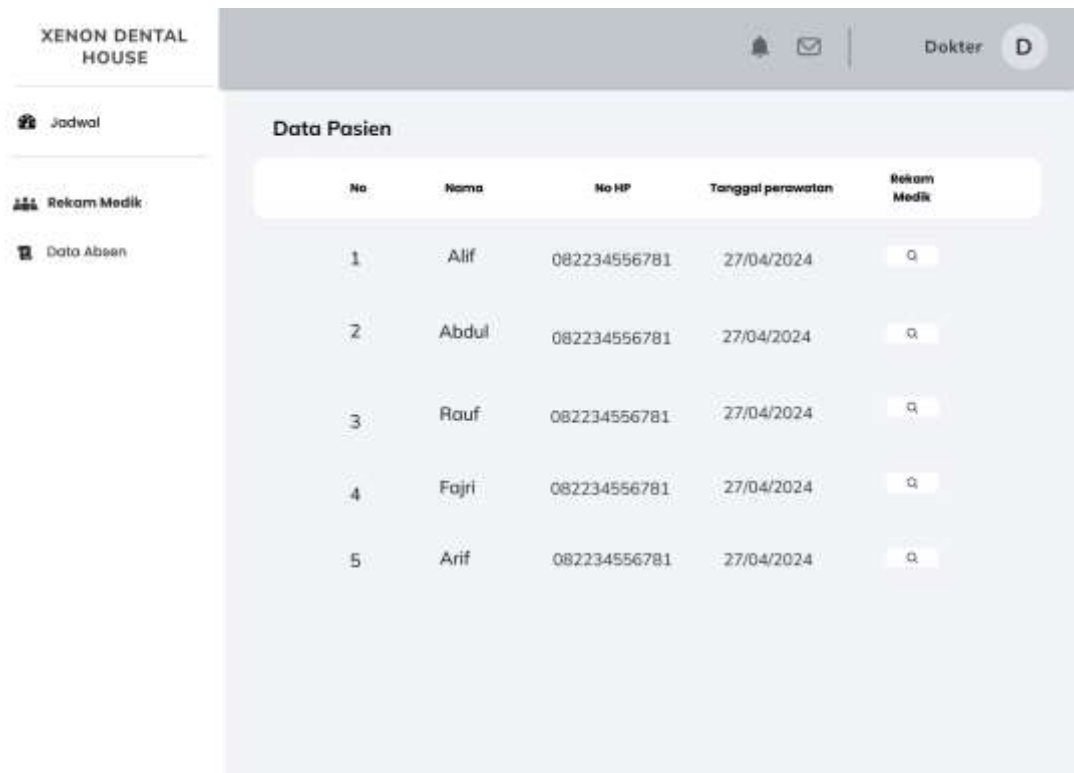
### Data Absen

Kembali

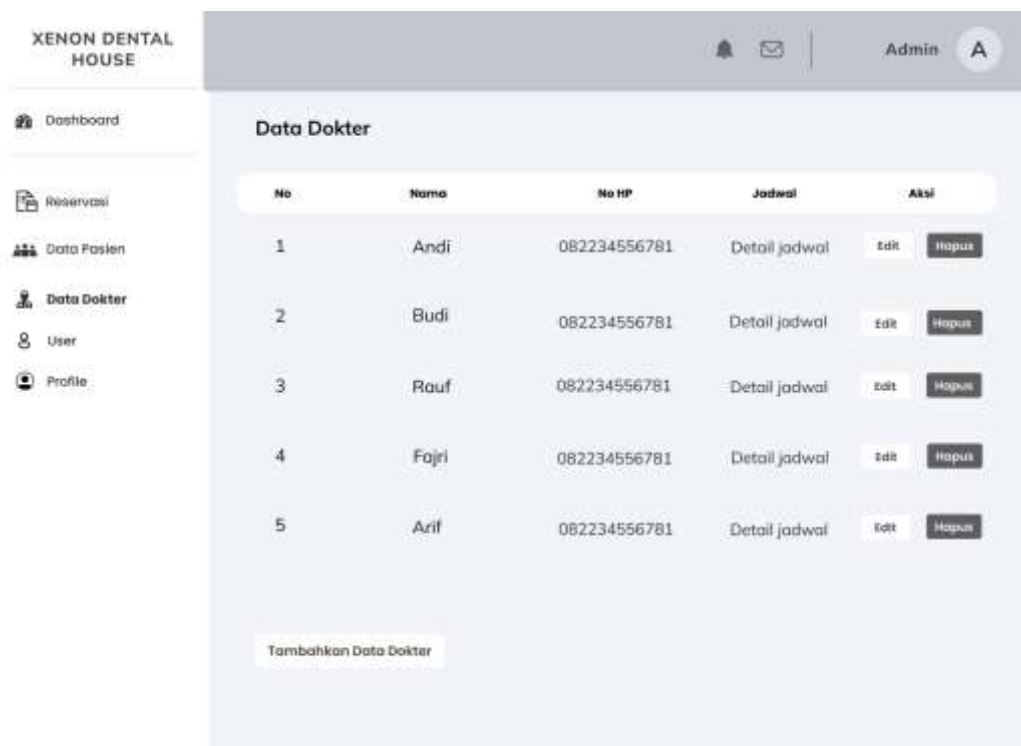
No	Tanggal	Alasan	Aksi	
1	27/04/2024	Sakit	Edit	Hapus
2	27/04/2024	Kemalangan	Edit	Hapus
3	27/04/2024	Sakit	Edit	Hapus
4	27/04/2024	Sakit	Edit	Hapus
5	27/04/2024	Cuti	Edit	Hapus

Tambahkan Data Absen

#### 14. Mockup Antarmuka Halaman Data Pasien User Dokter

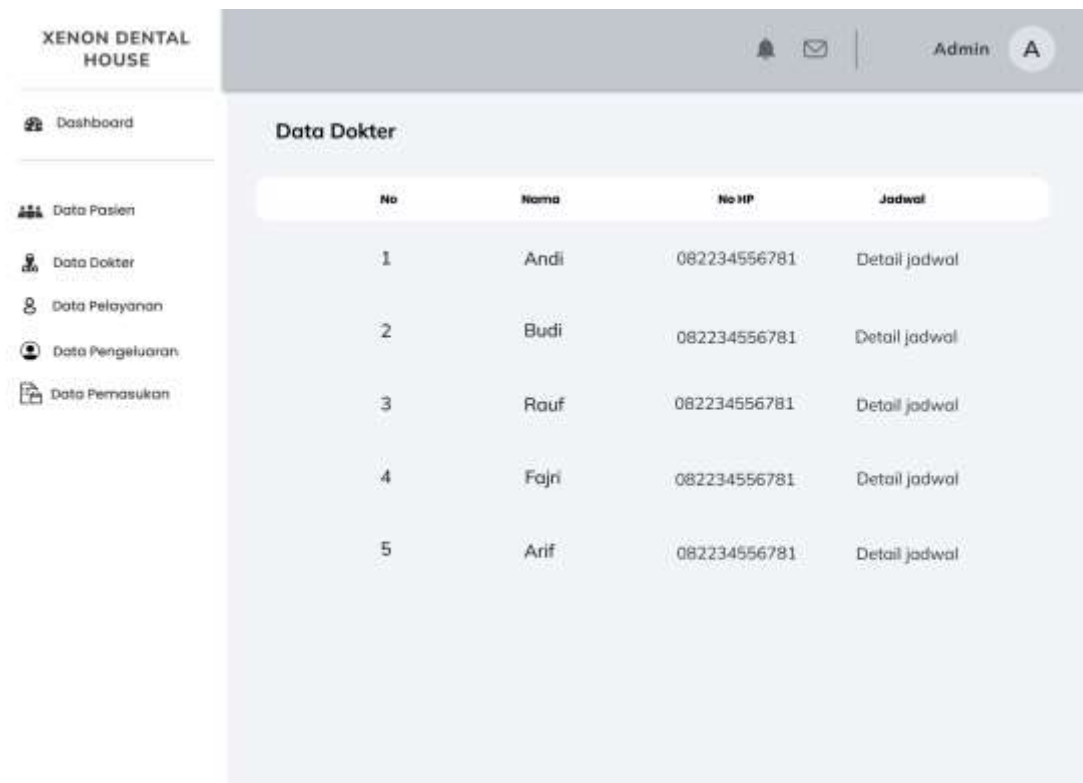


#### 15. Mockup Antarmuka Halaman Data Dokter User Admin

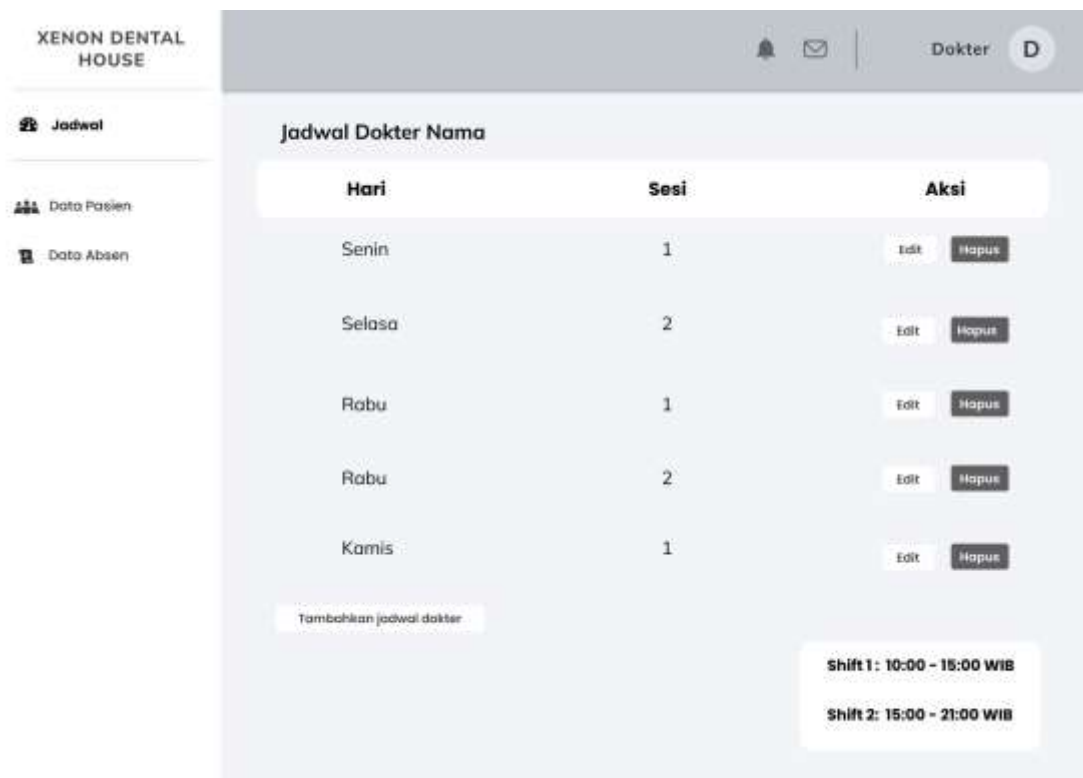




## 16. Mockup Antarmuka Halaman Data Dokter User Owner



## 17. Mockup Antarmuka Halaman Dabsboard Dokter



## 18. Mockup Antarmuka Halaman Rekam Medik User Dokter

The mockup shows a web interface for 'XENON DENTAL HOUSE'. On the left is a sidebar with navigation links: Dashboard, Reservasi, Data Pasien, Data Dokter, User, and Profile. The top header includes a notification bell, an email icon, and a user profile labeled 'Dokter D'. The main content area is titled 'Rekam Medik' and contains the following patient information: Nama : Alif, Tanggal lahir & Umur : 20, Perawatan, Riwayat, and Kondisi Pasien. Below this, the 'Lembar Tindakan' (Treatment Sheet) is displayed with the text 'S : Gigi Berlubang', 'O:', 'A', and 'P'. There are 'Kembali' (Back) and 'Edit' buttons.

**XENON DENTAL HOUSE**

Dashboard  
Reservasi  
Data Pasien  
Data Dokter  
User  
Profile

Rekam Medik

Kembali

Nama : Alif  
Tanggal lahir & Umur : 20  
Perawatan  
Riwayat  
Kondisi Pasien

Lembar Tindakan :  
S : Gigi Berlubang  
O:  
A  
P

Edit

## 19. Mockup Antarmuka Halaman Edit data pasien User Dokter

The mockup shows a web interface for 'XENON DENTAL HOUSE'. On the left is a sidebar with navigation links: Dashboard, Reservasi, Data Pasien, Data Dokter, User, and Profile. The top header includes a notification bell, an email icon, and a user profile labeled 'Dokter D'. The main content area is titled 'Edit Data Pasien' and contains input fields for Nama, Nomor HP, Alamat, Email, and Tanggal Penawatan. There is a 'Kembali' (Back) button and a 'Submit' button.

**XENON DENTAL HOUSE**

Dashboard  
Reservasi  
Data Pasien  
Data Dokter  
User  
Profile

Edit Data Pasien

Kembali

Nama  
Nomor HP  
Alamat  
Email  
Tanggal Penawatan

Submit

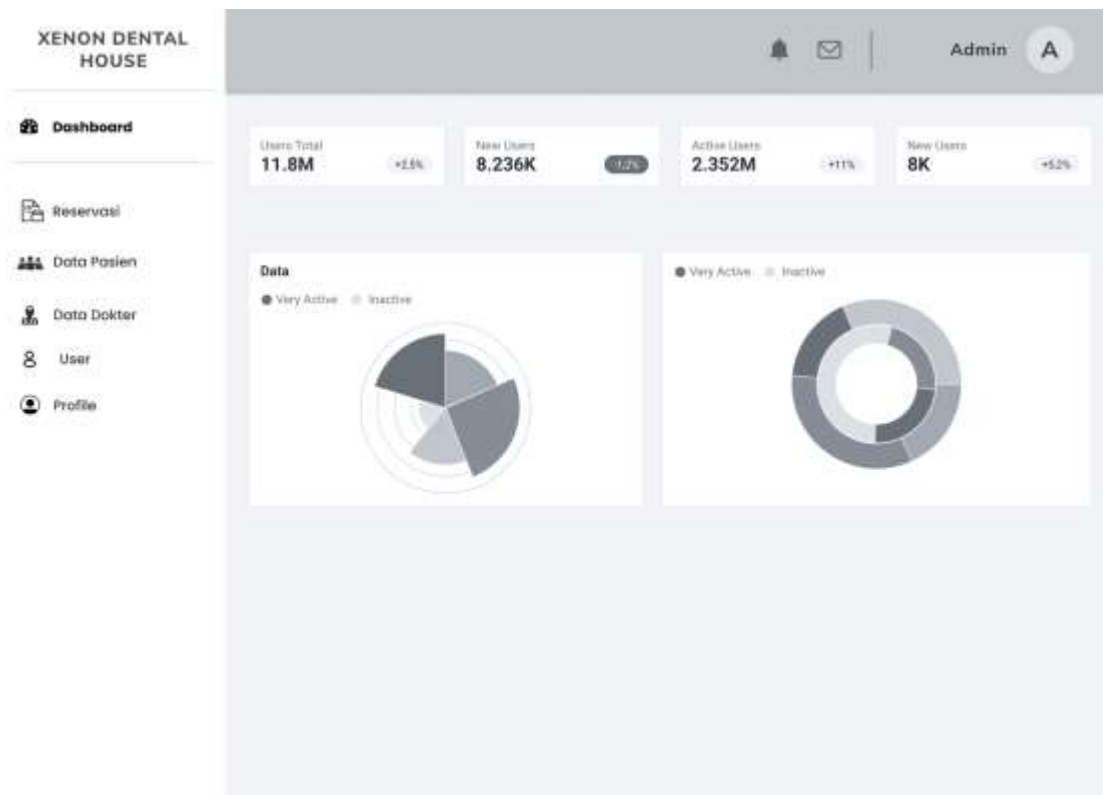
## 20. Mockup Antarmuka Halaman Tambah jadwal dokter

The mockup shows a web interface for 'XENON DENTAL HOUSE'. On the left is a sidebar menu with options: Dashboard, Reservasi, Data Pasien, Data Dokter, User, and Profile. The top header includes a notification bell, an email icon, and a user profile labeled 'Dokter D'. The main content area is titled 'Tambahkan Jadwal Dokter' and features a 'Kembali' (Back) button in the top right. The form contains two input fields labeled 'Hari' and 'Sesi', followed by a 'Submit' button. At the bottom right, there is a box displaying two shifts: 'Shift 1: 10:00 - 15:00 WIB' and 'Shift 2: 15:00 - 21:00 WIB'.

## 21. Mockup Antarmuka Halaman Edit Data dokter

The mockup shows a web interface for 'XENON DENTAL HOUSE'. The sidebar menu on the left includes: Jadwal, Rekam Medik, and Data Absen. The top header features a notification bell, an email icon, and a user profile labeled 'Dokter D'. The main content area is titled 'Edit Data Dokter' and includes a 'Kembali' (Back) button in the top right. The form has three input fields labeled 'Nama', 'Nomor Handphone', and 'Email', with a 'Submit' button below them.

## 22. Mockup Antarmuka Halaman Dashboard Admin



## 23. Mockup Antarmuka Halaman Tambah data pelayanan

**XENON DENTAL HOUSE**

Doktor **D**

**Tambahkan Data Pelayanan**

Kembali

Nama Pelayanan

Harga

Submit

**Dashboard**

- Data Pasien
- Data Dokter
- Data Pelayanan
- Data Pengeluaran
- Data Pemasukan

## 24. Mockup Antarmuka Halaman Data Pelayanan

XENON DENTAL HOUSE

Dashboard

Data Pasien

Data Dokter

Data Pelayanan

Data Pengeluaran

Data Pemasukan

Dokter D

Harga Pelayanan

No	Pelayanan	Harga	Aksi
1	Cabut Gigi	200.000	<div>Edit</div> <div>Hapus</div>
2	Scaling	200.000	<div>Edit</div> <div>Hapus</div>
3	Bleaching	100.000	<div>Edit</div> <div>Hapus</div>
4	Gigi Tiruan	200.000	<div>Edit</div> <div>Hapus</div>
5	Tambal Gigi	1000.000	<div>Edit</div> <div>Hapus</div>

Tambahkan Data Pelayanan

## 25. Mockup Antarmuka Halaman Pemasukan

XENON DENTAL HOUSE

Dashboard

Data Pasien

Data Dokter

Data Pelayanan

Data Pengeluaran

Data Pemasukan

Dokter D

Pemasukan

Pemasukan Bulan Ini

Pengeluaran Bulan Ini

Profit Bulan Ini

RP 100.000

RP 100.000

RP 100.000

No	Pemasukan	Tanggal	Profit
1	Cabut Gigi	200.000	<div>Edit</div> <div>Hapus</div>
2	Scaling	200.000	<div>Edit</div> <div>Hapus</div>
3	Bleaching	100.000	<div>Edit</div> <div>Hapus</div>
4	Gigi Tiruan	200.000	<div>Edit</div> <div>Hapus</div>
5	Tambal Gigi	1000.000	<div>Edit</div> <div>Hapus</div>

Tambahkan Data Pelayanan

## 26. Mockup Antarmuka Halaman Edit Pemasukan

**XENON DENTAL HOUSE**

Dashboard

Data Pasien

Data Dokter

Data Pelayanan

Data Pengeluaran

Data Pemasukan

**Edit Data Pemasukan** [Kembali](#)

Nama Pelayanan

Harga

Submit

## 27. Mockup Antarmuka Halaman Data Pengeluaran

**XENON DENTAL HOUSE**

Dashboard

Data Pasien

Data Dokter

Data Pelayanan

Data Pengeluaran

Data Pemasukan

**Pengeluaran**

Pemasukan Bulan Ini  
RP 100.000

Pengeluaran Bulan Ini  
RP 100.000


Profit Bulan Ini  
RP 100.000


No	Pengeluaran	Tanggal	Action
1	Cabut Gigi	200.000	<a href="#">Edit</a> <a href="#">Hapus</a>
2	Scaling	200.000	<a href="#">Edit</a> <a href="#">Hapus</a>
3	Bleaching	100.000	<a href="#">Edit</a> <a href="#">Hapus</a>
4	Gigi Tiruan	200.000	<a href="#">Edit</a> <a href="#">Hapus</a>
5	Tambal Gigi	1000.000	<a href="#">Edit</a> <a href="#">Hapus</a>


[Tambahkan Data Pelayanan](#)


## 28. Mockup Antarmuka Halaman Tambah Pengeluaran


**XENON DENTAL  
HOUSE**


 Dashboard



 Data Pasien


 Data Dokter

 Data Pelayanan

 Data Pengeluaran

 Data Pemasukan



Dokter 

### Tambah data Pengeluaran

Kembali

Nama Pelayanan:

Jumlah Pengeluaran:

Submit

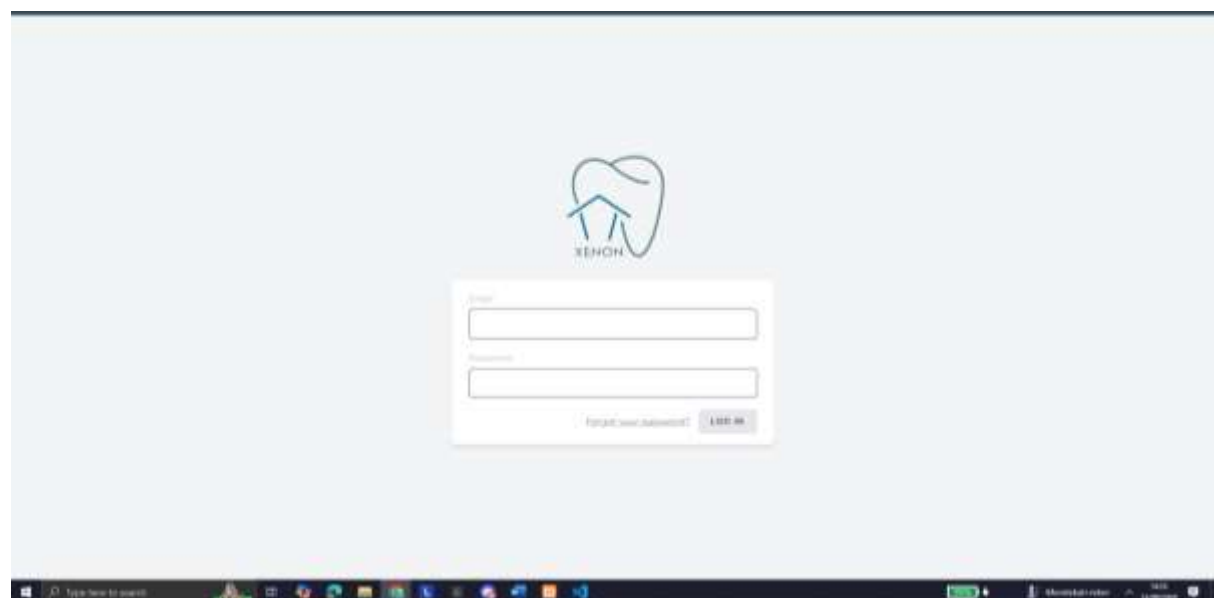
**LAMPIRAN D**  
**IMPLEMENTASI ANTARMUKA**  
**(LANJUTAN)**



## 1. Implementasi Antarmuka Landing Page



## 2. Implementasi Antarmuka Login User



### 3. Implementasi Antarmuka Ubah Password dan Hapus Akun User

The screenshot displays a web application interface with two main sections. The top section, titled 'Update Password', includes a sub-header 'Secure your account by using a strong, random password to stay secure.' and three input fields labeled 'Current Password', 'New Password', and 'Confirm Password'. A 'SAVE' button is positioned below these fields. The bottom section, titled 'Delete Account', features a sub-header 'Once your account is deleted, all of its resources and data will be permanently removed. Before deleting your account, please download any data or information that you wish to retain.' and a red 'DELETE ACCOUNT' button.

### 4. Implementasi Antarmuka Ubah Informasi Profile

The screenshot shows a web application interface with a 'Profile' header. Below the header, there are two sections. The first section, 'Profile Information', has a sub-header 'Update your account's profile information and email address.' and two input fields labeled 'Name' (containing 'John') and 'Email' (containing 'john@gmail.com'). A 'SAVE' button is located below these fields. The second section, 'Update Password', has a sub-header 'Secure your account by using a strong, random password to stay secure.' and a single input field labeled 'Current Password'.

## 5. Implementasi Antarmuka Tambah Reservasi (User Admin)

**KEMAH DENTAL HOUSE**

**Form Tambah Reservasi**

Nama:

Jenis Kelamin:

Tanggal Lahir:

Alamat:

No. Telepon:

Pelayanan (Pilih salah satu):

Tipe Layanan:

## 6. Implementasi Antarmuka Data Pasien (User Admin)

**KEMAH DENTAL HOUSE**

**Data Pasien**

Show: 7 entries

ID	Nama	Jenis Kelamin	Alamat	No. HP	Aksi
1	Budi Santoso	Pria	Jl. Merdeka No. 1, Jakarta	08123456789	<a href="#">Edit</a> <a href="#">Hapus</a>
2	Siti Aminah	Wanita	Jl. Asia Afrika No. 2, Bandung	08234567890	<a href="#">Edit</a> <a href="#">Hapus</a>
3	Rahmat	Pria	Jl. Sudirman No. 3, Surabaya	08345678901	<a href="#">Edit</a> <a href="#">Hapus</a>
4	Fitri	Wanita	Jl. Diponegoro No. 4, Semarang	08456789012	<a href="#">Edit</a> <a href="#">Hapus</a>
5	Andi	Pria	Jl. A Yani No. 5, Medan	08567890123	<a href="#">Edit</a> <a href="#">Hapus</a>
6	Rizki	Wanita	Jl. Veteran No. 6, Yogyakarta	08678901234	<a href="#">Edit</a> <a href="#">Hapus</a>
7	Dimas	Pria	Jl. Pahlawan No. 7, Makassar	08789012345	<a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 7 of 7 entries

Page 1 of 1

## 7. Implementasi Antarmuka Tambah Data Pasien (User Admin)

**Tambah data Pasien**

Nama Pasien

SS. Adhik

Telepon (Lahir)

Berkas

Tanggal Lahir

0805-10-24

Religi

Islam

Alamat

8 Jln. Artha Pda. 3, Bandung

No HP

08228871390

**Simpan**

Copyright © Kemon Dental House

## 8. Implementasi Antarmuka Rekam Medik (User Admin)

**Rekam Medik Siti Aminah**

Dokter	Dr.
Sejarah medis	AD
Tekanan darah	44
Penyakit diabetes	Tidak Ada
Dukungan	Tidak Ada
Hipertensi	Tidak Ada
Penyakit jantung	Tidak Ada
Gejala Malokasi	Tidak Ada
Gejala Gigi	Tidak Ada
Karies	Tidak Ada
Perawatan	(Berkas)
Biaya	00


**Simpan Rekam Medik**

Copyright © Kemon Dental House

The screenshot shows the 'Edit Rekam Medik' (Edit Medical Record) form. The left sidebar contains the application logo and a menu with options: Dashboard, Rekam Medik, Data Pasien, Data Rawat, and Data User. The main form area has the following fields:

- Idangian Dasar:** A text input field.
- Tekanan Darah:** A text input field.
- Persepsi Rongga:** A text input field.
- Palupulpa:** A text input field.
- Stomatop:** A text input field.
- Impaksi:** A text input field.
- Persepsi Lidah:** A text input field.
- Rongga Mulut:** A text input field.
- Rongga Teling:** A text input field.

At the bottom of the screen, there is a Windows taskbar with various icons and a system clock showing 10:00 on 11/04/2020.



KEMON  
DENTAL  
HOUSE

Dashboard

Keuangan

Data Pasien

Data Dokter

Data User

Logout

about

## Data Dokter

Show 10/1000

Search

No	Tg	Nama	No HP	Jadwal	Aksi
1		Dr. Andi Wijaya	029246799013	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
2		Adi	0252276539913	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
3		Rani	029239367718	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
4		Adi S.	029450672713	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
5		Baki	02645277361	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
6		Dani	029456723088	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>
7		Andi	0297542684643	Detail Jadwal	<a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 7 of 7 entries

Previous 1 Next

Copyright © Kemon Dental House

## 11. Implementasi Antarmuka Edit Data Dokter (User Admin)

**KENOM DENTAL HOUSE**

Dashboard Perawatan Data Pasien Data Dokter Data User

### Edit data Dokter

Nama Dokter: Dr. Andi Wijayan

Alamat: R. Kesehatan No. 3, Surabaya

Nomor HP: 081456789012

Simpan

Copyright © KENOM DENTAL HOUSE

## 12. Implementasi Antarmuka Detail Jadwal Dokter (User Admin)

**KENOM DENTAL HOUSE**

Dashboard Perawatan Data Pasien Data Dokter Data User

### Detail Jadwal Dokter

Search: [ ]

No	Dokter	Jam
1	Dokter	Jam

Showing 1 of 1 entries

Tambahkan Data Jadwal Dokter

SHR 1 : 10.00 - 15.00  
SHR 2 : 15.00 - 21.00

Copyright © KENOM DENTAL HOUSE

### 13. Implementasi Antarmuka Edit Jadwal Dokter (User Admin)

SENON DENTAL HOUSE

Dashboard

Reschedule

Data Pasien

Data Dokter

Data User

### Edit Jadwal Dokter

Hari:

Jam:

Simpan

Shift 1: 10.00 - 15.00  
Shift 2: 15.00 - 21.00

Copyright © Senon Dental House

### 14. Implementasi Antarmuka Tambah Jadwal Dokter (User Admin)

SENON DENTAL HOUSE

Dashboard

Reschedule

Data Pasien

Data Dokter

Data User

### Tambah Jadwal Dokter

Hari:

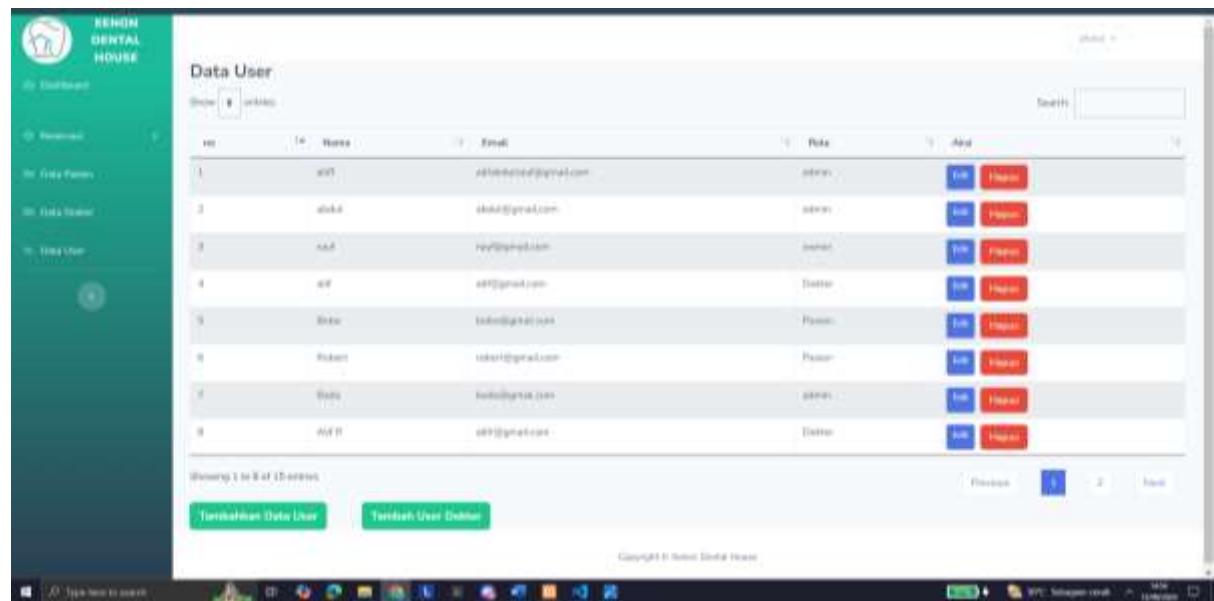
Jam:

Simpan

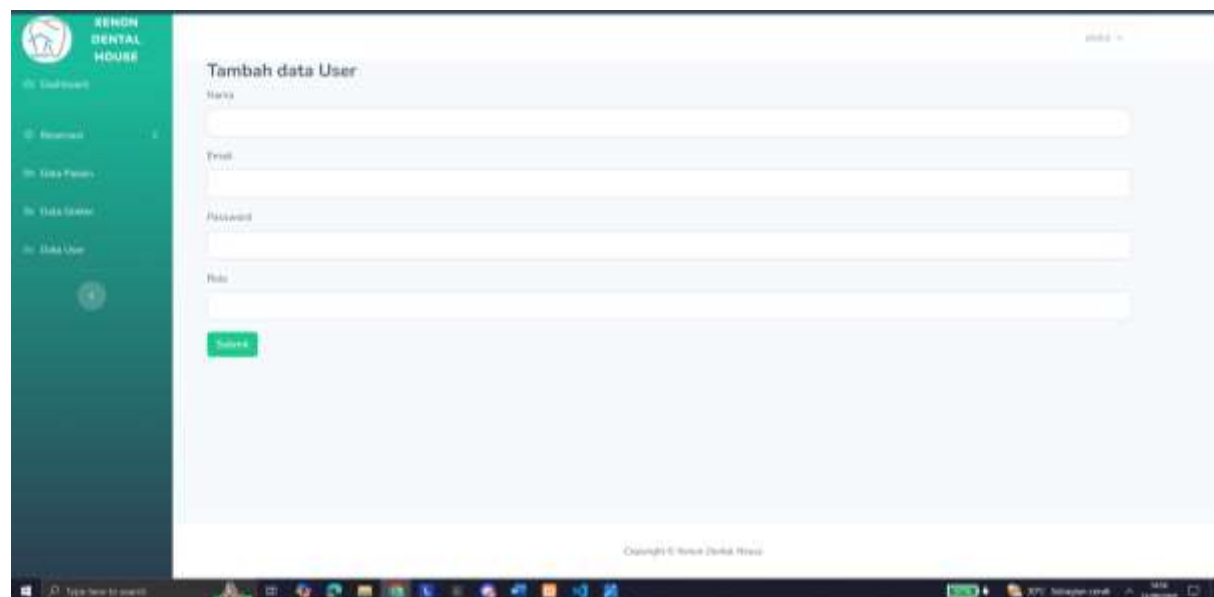
Shift 1: 10.00 - 15.00  
Shift 2: 15.00 - 21.00

Copyright © Senon Dental House

## 15. Implementasi Antarmuka Data User (User Admin)



## 16. Implementasi Antarmuka Tambah Data User (User Admin)





## 17. Implementasi Antarmuka Tambah Data Dokter (User Admin)

The screenshot shows the 'Tambah data Dokter' (Add Doctor Data) form in the XENON DENTAL HOUSE application. The form is located on the right side of the screen, with a green sidebar on the left containing navigation links: Dashboard, Resep, Data Pasien, Data Dokter, and Data User. The form fields are as follows:

- Nama Dokter:
- Alamat:
- Nomor Hp:
- Lokasi Praktik:
- Picking:
- Email:
- Password:

A green 'Simpan' (Save) button is located at the bottom of the form. The footer of the application shows 'Copyright © Xenon Dental House'.

## 18. Implementasi Antarmuka Edit Data User (User Admin)

The screenshot shows the 'Edit data User' form in the XENON DENTAL HOUSE application. The form is located on the right side of the screen, with a green sidebar on the left containing navigation links: Dashboard, Resep, Data Pasien, Data Dokter, and Data User. The form fields are as follows:

- Nama:
- no. Hp:
- Email:
- Role:

A green 'Simpan' (Save) button is located at the bottom of the form. The footer of the application shows 'Copyright © Xenon Dental House'.

## 19. Implementasi Antarmuka Registrasi Akun (User Pasien)

The screenshot shows a web application for 'XENON DENTAL HOUSE'. At the top center is a logo featuring a tooth with a house inside it, with the word 'XENON' below. Below the logo is a registration form with the following fields: 'Nama' (Name), 'Email', 'Password', and 'Confirm Password'. At the bottom of the form is a 'REGISTER' button. The form is set against a light blue background.

## 20. Implementasi Antarmuka History Reservasi (User Pasien)

The screenshot shows the 'History Reservasi' (Reservation History) page of the XENON Dental House application. The page has a green sidebar on the left with the application logo and navigation links. The main content area displays a table of reservation history with the following columns: No, Nama Pasien, Lokasi Perawatan, Nama Dokter, Tanggal, Jam, and Status. There are two rows of data. Below the table, it says 'Showing 1 to 2 of 2 entries'. At the bottom right, there are 'Previous', '1', and 'Next' buttons.

No	Nama Pasien	Lokasi Perawatan	Nama Dokter	Tanggal	Jam	Status
1	Robert	Paling	Dr. Jodi Wijaya	2024-07-23	11:00:00	Belum
2	Robert	Paling	dr	2024-07-24	11:00:00	Belum

## 21. Implementasi Antarmuka Tambah Reservasi (User Pasien)

**Form Tambah Reservasi**

Name

Appointment Date

Appointment Time

Appointment Location

Appointment Type

Appointment Status

Appointment Reason

Appointment Notes

Appointment Location

## 22. Implementasi Antarmuka Jadwal Dokter (User Dokter)

**Jadwal Saya**

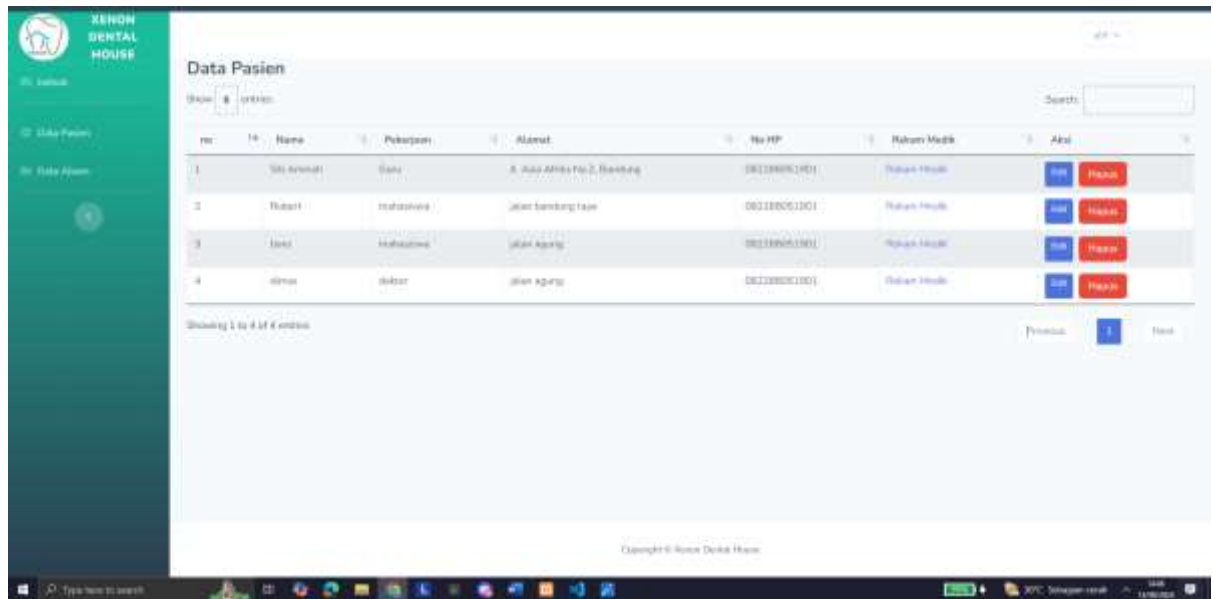
Search

No	TR	Hari	Jam
1		Senin	1
2		Selasa	2

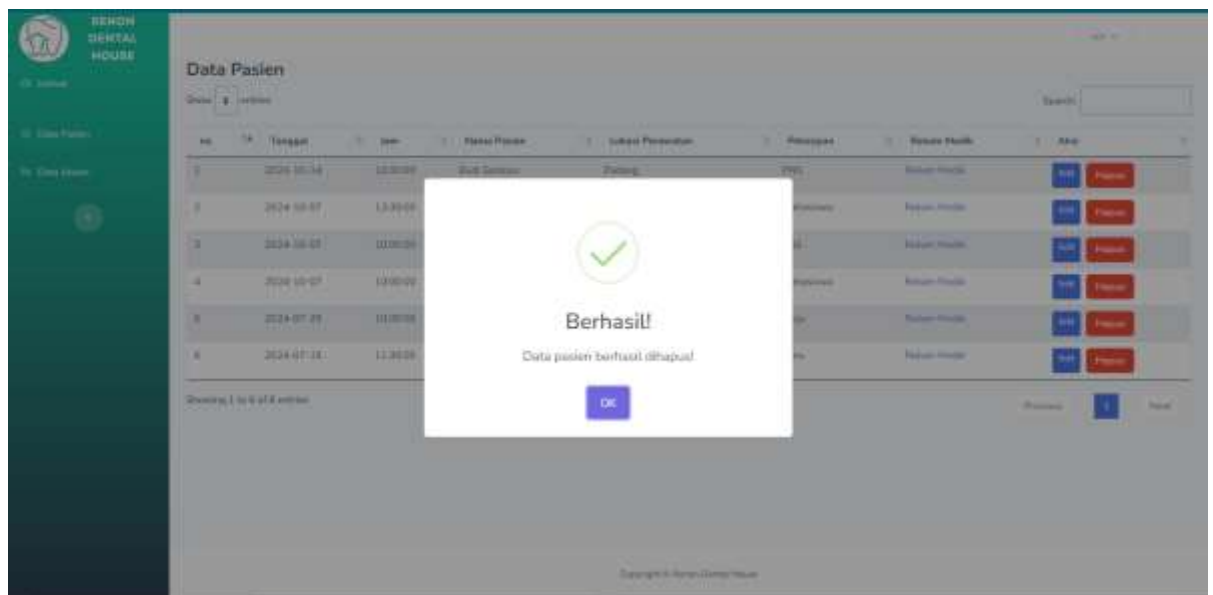
Showing 1 to 2 of 2 entries

Previous 1 Next

## 23. Implementasi Antarmuka Data Pasien (User Dokter)



## 24. Implementasi Antarmuka Hapus Data Pasien (User Dokter)



## 25. Implementasi Antarmuka Edit Data Pasien (User Dokter)

**Edit data Pasien**

Name Pasien

SA Anwar

Telephone

Bandung

Birthdate

1995-08-24

Gender

Male

Address

4. Aya Mulya No. 1, Bandung


No. HP

081234567890

[Save](#)

Copyright © XENON DENTAL HOUSE

## 26. Implementasi Antarmuka Data Perizinan (User Dokter)



XENON  
DENTAL  
HOUSE

Dashboard

Daftar Pasien

Daftar Apoin

Data Perizinan

5 entries

Search:

No	No	Tanggal Awal	Tanggal Akhir	Alamat	Status	Aksi
1	2024-06-05	2024-06-10	Utaman	Dikirim	<div><div>Edit</div><div>Hapus</div></div>	
2	2024-06-15	2024-06-20	Potatbat	Dikirim	<div><div>Edit</div><div>Hapus</div></div>	
3	2024-07-04	2024-07-11	bandung sakit	Dikirim	<div><div>Edit</div><div>Hapus</div></div>	
4	2024-07-29	2024-08-02	bandung ke luar kota	Dikirim	<div><div>Edit</div><div>Hapus</div></div>	
5	2024-07-26	2024-07-31	sakit	Dikirim	<div><div>Edit</div><div>Hapus</div></div>	

Showing 5 to 5 of 5 entries

Previous

1

Next

Tambahkan Data Baru

Copyright © XENON DENTAL HOUSE

## 27. Implementasi Antarmuka Tambah Data Absen (User Dokter)

**Tambah data Absen**

Tanggal Absen

Tanggal Refer

Waktu

**Simpan**

## 28. Implementasi Antarmuka Data Pasien (User Dokter)

**Data Pasien**

Show

ID	Tanggal	Jam	Nama Pasien	Lokasi Pemeriksaan	Referral	Referral Date	Aksi
1	2024-10-14	10:00:00	Budi Santoso	Paling	PHG	<a href="#">Referral</a>	<a href="#">Edit</a> <a href="#">Hapus</a>
2	2024-10-07	13:30:00	Dimas	Paling	mahasiswa	Paling	<a href="#">Edit</a> <a href="#">Hapus</a>
3	2024-10-07	10:00:00	Budi Santoso	Paling	PHG	<a href="#">Referral</a>	<a href="#">Edit</a> <a href="#">Hapus</a>
4	2024-10-07	10:00:00	Ulf Azzul	Paling	mahasiswa	<a href="#">Referral</a>	<a href="#">Edit</a> <a href="#">Hapus</a>
5	2024-07-28	10:00:00	Rizki	Paling	Karya	<a href="#">Referral</a>	<a href="#">Edit</a> <a href="#">Hapus</a>
6	2024-07-15	11:30:00	Si Azzul	Paling	Clara	<a href="#">Referral</a>	<a href="#">Edit</a> <a href="#">Hapus</a>

Showing 1 to 6 of 6 entries

Previous **1** Next

## 29. Implementasi Antarmuka Edit Data Pasien (User Dokter)

The screenshot shows the 'Edit data Pasien' form in the RSMH DENTAL HOUSE application. The form is titled 'Edit data Pasien' and contains several input fields for patient information. The left sidebar shows the navigation menu with options: Dashboard, RSMH, Data Pasien, Data Dokter, and Data User. The form fields are as follows:

Field	Value
Nama Pasien	Si Anwar
Tanggal Lahir	1995-08-24
Gender	Pria
Alamat	4 Jln. Mulya No. 1, Bandung
No. HP	082288888888

At the bottom of the form is a blue 'Simpan' button. The footer of the application shows the text 'RSMH DENTAL HOUSE'.

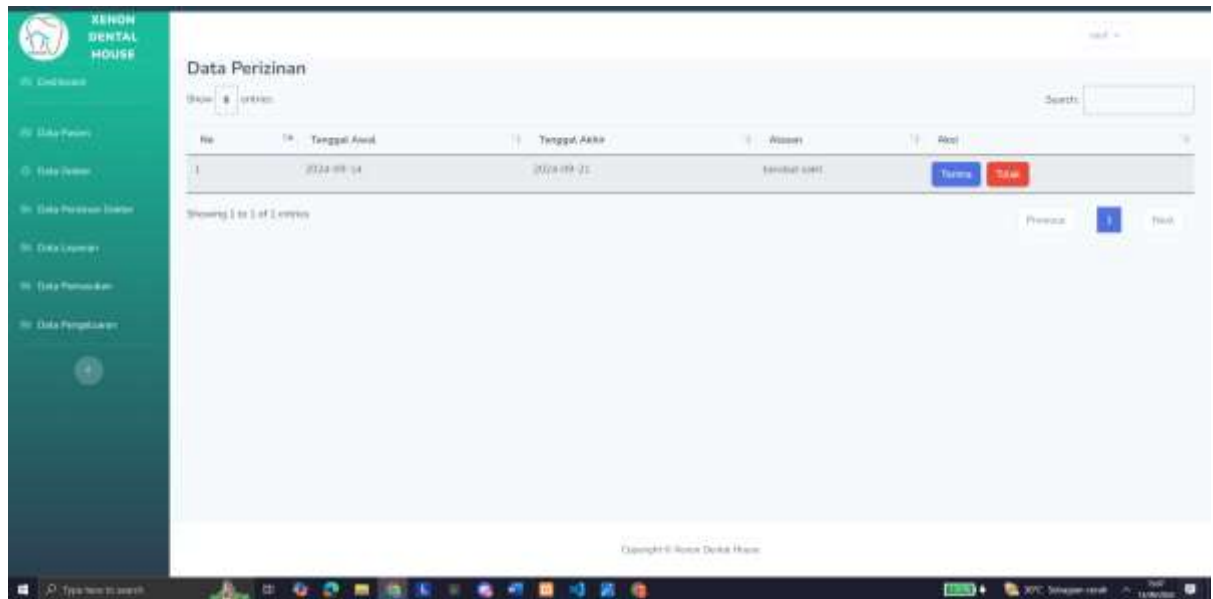
## 30. Implementasi Antarmuka Rekam Medik (User Owner)

The screenshot shows the 'Rekam Medik Budi Santoso (2024-10-14)' form in the RSMH DENTAL HOUSE application. The form is titled 'Rekam Medik Budi Santoso (2024-10-14)' and contains a list of medical history items. The left sidebar shows the navigation menu with options: Dashboard, RSMH, Data Pasien, and Data Dokter. The form fields are as follows:

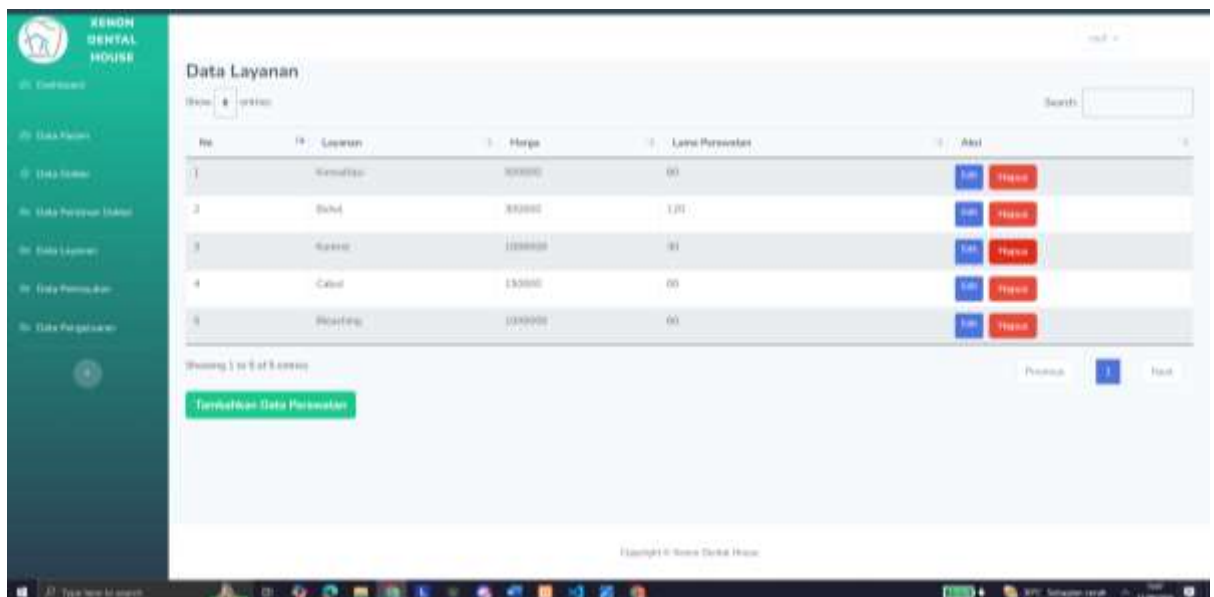
Field	Value
Dokter	Dr. A
Rekam medis	10
Tekanan darah	120 mmHg
Persentase lemak	Tinggi Ada
Glukosa	Tinggi Ada
Hemoglobin	Tinggi Ada
Persentase lemak	Tinggi Ada
Alergi Makanan	Tinggi Ada
Alergi Obat	Tinggi Ada
Saluran	Salut pingsang
Pemeriksaan	Bahut
Uji	1
Rekamasi Tindakan Lain/Rekam	Rekamasi Tindakan Lain/Rekam

At the bottom of the form is a green 'Edit Rekam Medik' button. The footer of the application shows the text 'RSMH DENTAL HOUSE'.

### 31. Implementasi Antarmuka List Perizinan Dokter (User Owner)



### 32. Implementasi Antarmuka Data Layanan (User Owner)





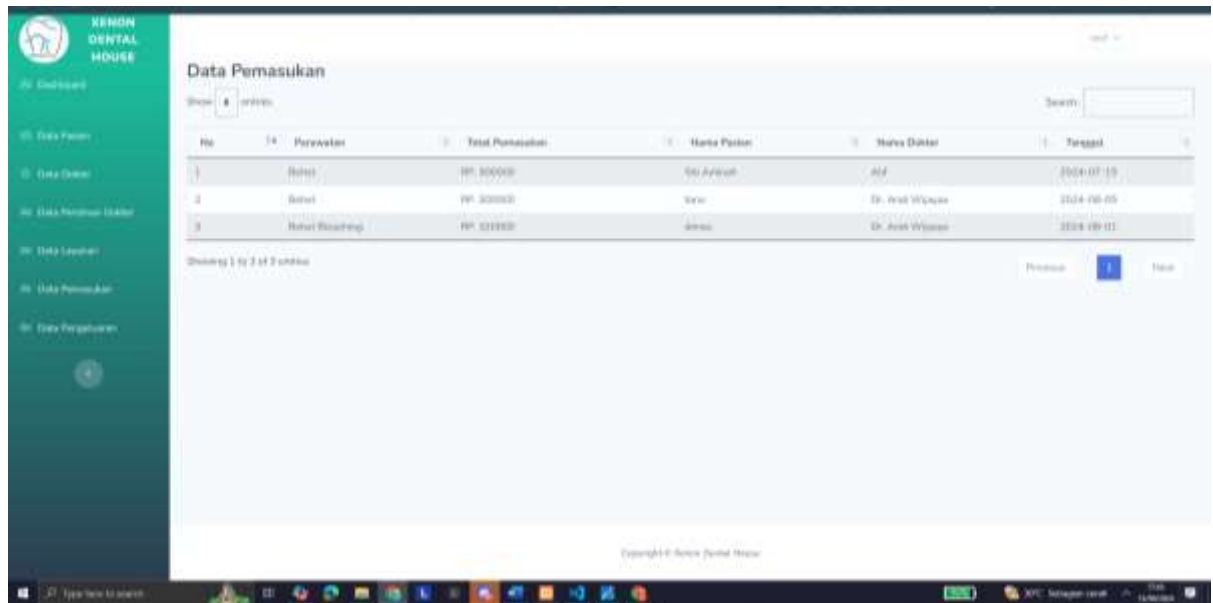
### 33. Implementasi Antarmuka Tambah Data Layanan (User Owner)

The screenshot shows the 'Tambah data Layanan' (Add Service Data) form. On the left is a green sidebar with the application logo and a list of menu items: Dashboard, Data Pasien, Data Dokter, Data Perawatan, Data Layanan, Data Penjualan, and Data Pengeluaran. The main area contains the form with the following fields: 'Layanan' (Service), 'Harga' (Price), and 'Estimasi Waktu' (Estimated Time). Each field has a corresponding input box. Below the 'Estimasi Waktu' field is a green 'Simpan' (Save) button. The footer of the application is visible at the bottom, showing the text 'Copyright © Kenon Dental House'.

### 34. Implementasi Antarmuka Edit Layanan (User Owner)

The screenshot shows the 'Edit data Layanan' (Edit Service Data) form. The layout is identical to the 'Tambah data Layanan' form, with a green sidebar on the left and a main form area on the right. The form fields are 'Layanan', 'Harga', and 'Estimasi Waktu', each with an input box. The 'Simpan' (Save) button is green and located below the 'Estimasi Waktu' field. The footer text 'Copyright © Kenon Dental House' is visible at the bottom.

### 35. Implementasi Antarmuka Data Pemasukan (User Owner)



**Data Pemasukan**

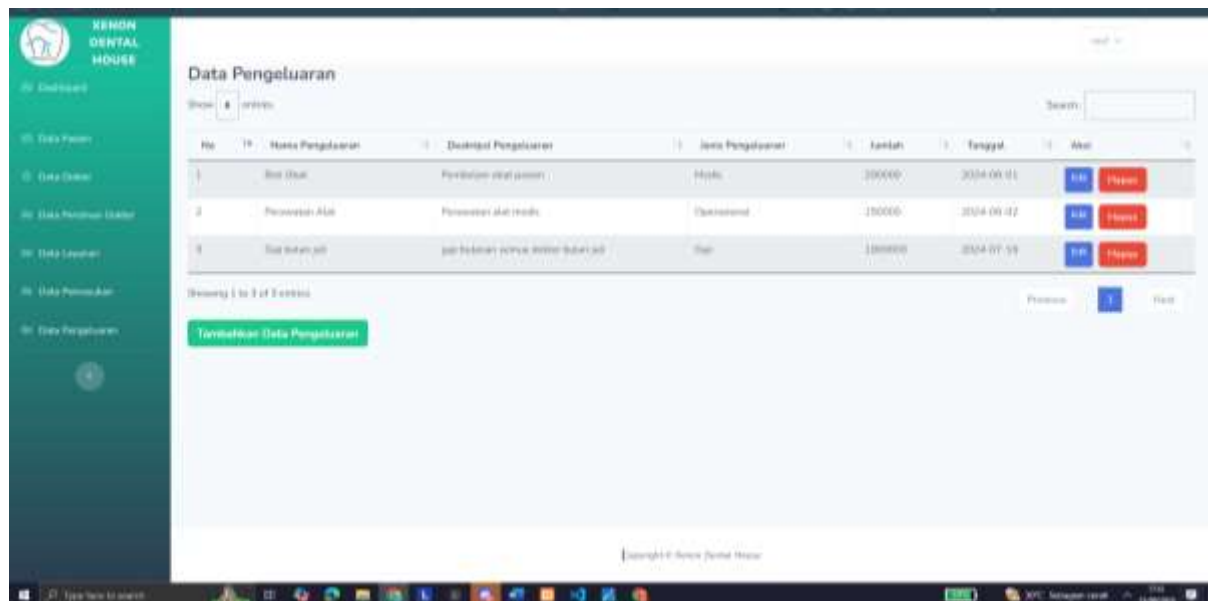
Show 3 entries

No	TR	Perawatan	Total Pemasukan	Nama Pasien	Nama Dokter	Tanggal
1		Retensi	RP. 300000	Mr. Ahmad	Adi	2024-07-15
2		Retensi	RP. 300000	Toni	Dr. Andi Wijaya	2024-08-05
3		Retensi Rencanggih	RP. 321950	Amos	Dr. Andi Wijaya	2024-08-01

Showing 3 to 3 of 3 entries

Processed 1 Read

### 36. Implementasi Antarmuka Data Pengeluaran (User Owner)



**Data Pengeluaran**

Show 3 entries

No	TR	Nama Pengeluaran	Deskripsi Pengeluaran	Jenis Pengeluaran	Jumlah	Tanggal	Aksi
1		Rent Ruang	Pembelian alat praktik	Modal	200000	2024-08-01	<a href="#">Edit</a> <a href="#">Delete</a>
2		Pembelian Alat	Pembelian alat praktik	Operasional	100000	2024-08-02	<a href="#">Edit</a> <a href="#">Delete</a>
3		Rent Ruang	gaji Praktikan selama praktik selama 300	Gaji	1000000	2024-07-19	<a href="#">Edit</a> <a href="#">Delete</a>

Showing 3 to 3 of 3 entries

Processed 1 Read

[Tambahkan Data Pengeluaran](#)

### 37. Implementasi Antarmuka Tambah Data Pengeluaran (User Owner)

The screenshot shows the 'Tambah data Pengeluaran' (Add Expense) form. The form is titled 'Tambah data Pengeluaran' and contains the following fields:

- Nama Pengeluaran: [Empty text input field]
- Deskripsi Pengeluaran: [Empty text input field]
- Jenis Pengeluaran: [Empty text input field]
- Jumlah Pengeluaran: [Empty text input field]
- Tanggal: [Date picker showing 01/06/2024]

At the bottom of the form is a blue 'Simpan' (Save) button. The left sidebar shows the application menu with 'Data Pengeluaran' selected. The footer displays 'Copyright © Xenon Dental House'.

### 38. Implementasi Antarmuka Edit Data Pengeluaran (User Owner)

The screenshot shows the 'Edit data Pengeluaran' (Edit Expense) form. The form is titled 'Edit data Pengeluaran' and contains the following fields:

- Nama Pengeluaran: [Text input field containing 'Beli Obat...']
- Deskripsi Pengeluaran: [Text input field containing 'Pembelian obat pasien']
- Jenis Pengeluaran: [Text input field containing 'Medis']
- Jumlah Pengeluaran: [Text input field containing '200000']
- Tanggal: [Date picker showing 01/06/2024]

At the bottom of the form is a blue 'Simpan' (Save) button. The left sidebar shows the application menu with 'Data Pengeluaran' selected. The footer displays 'Copyright © Xenon Dental House'.

**LAMPIRAN E**  
**KODE PROGRAM**  
**(LANJUTAN)**

## Kode Route

### 1. Auth.php

```
<?php

use App\Http\Controllers\Auth\AuthenticatedSessionController;
use App\Http\Controllers\Auth\ConfirmablePasswordController;
use App\Http\Controllers\Auth>EmailVerificationNotificationController;
use App\Http\Controllers\Auth>EmailVerificationPromptController;
use App\Http\Controllers\Auth\NewPasswordController;
use App\Http\Controllers\Auth>PasswordController;
use App\Http\Controllers\Auth>PasswordResetLinkController;
use App\Http\Controllers\Auth\RegisteredUserController;
use App\Http\Controllers\Auth\VerifyEmailController;
use Illuminate\Support\Facades\Route;

Route::middleware('guest')->group(function () {
    Route::get('register', [RegisteredUserController::class, 'create'])
        ->name('register');

    Route::post('register', [RegisteredUserController::class, 'store']);

    Route::get('login', [AuthenticatedSessionController::class, 'create'])
        ->name('login');

    Route::post('login', [AuthenticatedSessionController::class, 'store']);

    Route::get('forgot-password', [PasswordResetLinkController::class, 'create'])
        ->name('password.request');

    Route::post('forgot-password', [PasswordResetLinkController::class, 'store'])
        ->name('password.email');

    Route::get('reset-password/{token}', [NewPasswordController::class, 'create'])
        ->name('password.reset');

    Route::post('reset-password', [NewPasswordController::class, 'store'])
        ->name('password.store');
});

Route::middleware('auth')->group(function () {
    Route::get('verify-email', EmailVerificationPromptController::class)
        ->name('verification.notice');

    Route::get('verify-email/{id}/{hash}', VerifyEmailController::class)
        ->middleware(['signed', 'throttle:6,1'])
        ->name('verification.verify');

    Route::post('email/verification-notification',
[EmailVerificationNotificationController::class, 'store'])
        ->middleware('throttle:6,1')
        ->name('verification.send');

    Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
        ->name('password.confirm');

    Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);

    Route::put('password', [PasswordController::class, 'update'])->name('password.update');

    Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
        ->name('logout');
});
```

### 2. Web.php

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\ProfileController;
use App\Http\Controllers\AbsenceController;
use App\Http\Controllers\Admin\DataDokterController;
use App\Http\Controllers\Admin\HomeController;
use App\Http\Controllers\Dokter\IzinController;
```

```

use App\Http\Controllers\Admin\PasienController;
use App\Http\Controllers\Admin\UserController;
use App\Http\Controllers\Admin\JadwalController;
use App\Http\Controllers\Admin\RekamController;
use App\Http\Controllers\Admin\ReservasiController;
use App\Http\Controllers\Dokter\DokterHomeController;
use App\Http\Controllers\Owner\PerawatanController;
use App\Http\Controllers\Owner\PengeluaranController;
use App\Http\Controllers\Owner\PemasukanController;
use App\Http\Controllers\Pasien\ReservasiPasienController;
use App\Http\Controllers\WhatsAppController;

use App\Models\Perawatan;

require __DIR__.'/auth.php';

Route::get('/welcome', function () {
    return view('welcome');
});

Route::get('/manifest.json', function () {
    return response()->file(public_path('manifest.json'));
});

Route::get('/serviceworker.js', function () {
    return response()->file(public_path('serviceworker.js'));
});

Route::post('/send-whatsapp', [WhatsAppController::class, 'sendWhatsAppMessage'])->
    name('whatsapp.send');

Route::middleware('auth')->group(function () {
    Route::get('/home', function () {
        return view('landing_page');
    });
    Route::get('/pasien/history', [ReservasiPasienController::class, 'index'])->
        name('pasien.history');
    Route::get('/pasien/add', [ReservasiController::class, 'add'])->name('pasien.add');
    Route::post('/pasien/insert', [ReservasiPasienController::class, 'insert'])->
        name('pasien.insert');
    Route::get('/api/dokter-by-lokasi-p', [ReservasiPasienController::class,
        'getDokterByLokasi']);
    Route::get('/api/unavailable-times-p', [ReservasiPasienController::class,
        'getUnavailableTimes']);
    Route::get('/api/available-dates-p', [ReservasiPasienController::class,
        'getAvailableDates']);
    Route::get('/api/available-days-p', [ReservasiPasienController::class,
        'getAvailableDays']);
    Route::get('/api/booked-times-p', [ReservasiPasienController::class, 'getBookedTimes']);
    Route::get('/api/sesi-by-dokter-and-hari-p', [ReservasiPasienController::class,
        'getSesiByDokterAndHari']);

    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

Route::get('/', function () {
    return view('landing_page');
});
Route::get('/offline', function () {
    return view('offline');
});

Route::middleware(['auth', 'verified'])->group(function () {
    Route::get('/admin', [HomeController::class, 'index'])->name('admin');
    Route::post('/admin/filter', [HomeController::class, 'filter'])->name('home.filter');
});

Route::middleware(['role:admin'])->group(function () {
    // Routes untuk DataDokter
    Route::get('/datadokter', [DataDokterController::class, 'index'])->
        name('datadokter.index');
    Route::get('/datadokter/add', [DataDokterController::class, 'add'])->
        name('datadokter.add');
});

```

```

Route::post('/datadokter/insert', [DataDokterController::class, 'insert'])->
>name('datadokter.insert');
Route::get('/datadokter/edit/{id}', [DataDokterController::class, 'edit'])->
>name('datadokter.edit');
Route::post('/datadokter/update/{id}', [DataDokterController::class, 'update'])->
>name('datadokter.update');
Route::delete('/datadokter/destroy/{id}', [DataDokterController::class, 'destroy'])->
>name('datadokter.destroy');

// Routes untuk Jadwal DataDokter
Route::get('/datadokter/jadwal/{id}', [JadwalController::class, 'index'])->
>name('dokterjadwal.index');
Route::get('/datadokter/jadwal/add/{id}', [JadwalController::class, 'add'])->
>name('dokterjadwal.add');
Route::post('/datadokter/jadwal/insert/{id}', [JadwalController::class, 'insert'])->
>name('dokterjadwal.insert');
Route::get('/datadokter/jadwal/edit/{id}', [JadwalController::class, 'edit'])->
>name('dokterjadwal.edit');
Route::post('/datadokter/jadwal/update/{id}', [JadwalController::class, 'update'])->
>name('dokterjadwal.update');
Route::delete('/datadokter/jadwal/destroy/{id}', [JadwalController::class, 'destroy'])->
>name('dokterjadwal.destroy');

// Routes untuk Reservasi Admin
Route::get('/reservasi/admin/', [ReservasiController::class, 'index'])->
>name('reservasi.index');
Route::get('/reservasi/admin/add/', [ReservasiController::class, 'add'])->
>name('reservasi.add');
Route::post('/reservasi/admin/insert/', [ReservasiController::class, 'insert'])->
>name('reservasi.insert');
Route::get('/reservasi/admin/edit/{id}', [ReservasiController::class, 'edit'])->
>name('reservasi.edit');
Route::post('/reservasi/admin/update/{id}', [ReservasiController::class, 'update'])->
>name('reservasi.update');
Route::delete('/reservasi/admin/destroy/{id}', [ReservasiController::class, 'destroy'])->
>name('reservasi.destroy');
Route::post('/reservasi/admin/terima/{id}', [ReservasiController::class, 'terima'])->
>name('reservasi.terima');
Route::post('/reservasi/admin/tolak/{id}', [ReservasiController::class, 'tolak'])->
>name('reservasi.tolak');
Route::get('/reservasi/pasien/', [ReservasiController::class, 'pasien'])->
>name('reservasi.pasien');
Route::get('/api/dokter-by-lokasi', [ReservasiController::class, 'getDokterByLokasi']);
Route::get('/api/unavailable-times', [ReservasiController::class, 'getUnavailableTimes']);
Route::get('/api/available-dates', [ReservasiController::class, 'getAvailableDates']);
Route::get('/api/available-days', [ReservasiController::class, 'getAvailableDays']);
Route::get('/api/booked-times', [ReservasiController::class, 'getBookedTimes']);

Route::get('/api/sesi-by-dokter-and-hari', [ReservasiController::class,
'getSesiByDokterAndHari']);
Route::get('/reservasi/getDoctors', [ReservasiController::class, 'getDokterByLokasi'])->
>name('reservasi.getDoctors');
Route::get('/reservasi/getTimeSlots', [ReservasiController::class, 'getTimeSlots'])->
>name('reservasi.getTimeSlots');

// Routes untuk DataPasien
Route::get('/adatapasien', [PasienController::class, 'index'])->name('datapasien.index');
Route::get('/adatapasien/add', [PasienController::class, 'add'])->name('datapasien.add');
Route::post('/adatapasien/insert', [PasienController::class, 'insert'])->
>name('datapasien.insert');
Route::get('/adatapasien/edit/{id}', [PasienController::class, 'edit'])->
>name('datapasien.edit');
Route::post('/adatapasien/update/{id}', [PasienController::class, 'update'])->
>name('datapasien.update');
Route::delete('/adatapasien/destroy/{id}', [PasienController::class, 'destroy'])->
>name('datapasien.destroy');

// Routes untuk Rekam Pasien
Route::get('/datapasien/rekam/{id}', [RekamController::class, 'index'])->
>name('admin.rekam pasien');
Route::get('/datapasien/rekam/edit/{id}', [RekamController::class, 'edit'])->
>name('admin.rekam pasien.edit');
Route::post('/datapasien/rekam/update/{id}', [RekamController::class, 'update'])->
>name('admin.rekam pasien.update');

// Routes untuk DataUser
Route::get('/datauser', [UserController::class, 'index'])->name('datauser.index');
Route::get('/datauser/add', [UserController::class, 'add'])->name('datauser.add');

```

```

Route::post('/datauser/insert', [UserController::class, 'insert'])-
>name('datauser.insert');
Route::get('/datauser/edit/{id}', [UserController::class, 'edit'])->name('datauser.edit');
Route::post('/datauser/update/{id}', [UserController::class, 'update'])-
>name('datauser.update');
Route::delete('/datauser/destroy/{id}', [UserController::class, 'destroy'])-
>name('datauser.destroy');

Route::get('/datauserdokter/add', [UserController::class, 'add_dokter'])-
>name('datauserdokter.add');
Route::post('/datauserdokter/insert', [UserController::class, 'insert_dokter'])-
>name('datauserdokter.insert');
});

Route::get('/profil', function () {
    return view('admin.profil');
});

// Routes untuk dokter

Route::middleware(['role:Dokter'])->group(function () {
    Route::get('/dokter', [DokterHomeController::class, 'index'])->name('dokter.index');
    Route::get('/dataabsen', [IzinController::class, 'index'])->name('dataabsen.index');
    Route::get('/dataabsen/add', [IzinController::class, 'add'])->name('dataabsen.add');
    Route::get('/dataabsen/edit/{id}', [IzinController::class, 'edit'])->name('dataabsen.edit');
    Route::post('/dataabsen/update{id}', [IzinController::class, 'update'])-
>name('dataabsen.update');
    Route::post('/dataabsen/insert', [IzinController::class, 'insert'])-
>name('dataabsen.insert');
    Route::delete('/dataabsen/destroy{id}', [IzinController::class, 'destroy'])-
>name('dataabsen.destroy');

    Route::get('/ddatapasien', [PasienController::class, 'index'])->name('ddatapasien.index');
    Route::get('/ddatapasien/add', [PasienController::class, 'add'])->name('ddatapasien.add');
    Route::post('/ddatapasien/insert', [PasienController::class, 'insert'])-
>name('ddatapasien.insert');
    Route::get('/ddatapasien/edit/{id}', [PasienController::class, 'edit'])-
>name('ddatapasien.edit');
    Route::post('/ddatapasien/update/{id}', [PasienController::class, 'update'])-
>name('ddatapasien.update');
    Route::delete('/ddatapasien/destroy{id}', [PasienController::class, 'destroy'])-
>name('ddatapasien.destroy');
    Route::get('/ddatapasien/rekam/{id}', [RekamController::class, 'index'])-
>name('dokter.rekam pasien');
    Route::get('/ddatapasien/rekam/edit/{id}', [RekamController::class, 'edit'])-
>name('dokter.rekam pasien.edit');
    Route::post('/ddatapasien/rekam/update/{id}', [RekamController::class, 'update'])-
>name('dokter.rekam pasien.update');
});

Route::get('/kalender', function () {
    return view('absence');
});

// Route::resource('absences', AbsenceController::class);

Route::middleware(['role:owner'])->group(function () {
    // Routes untuk owner
    // Route::get('/owner', function () {
    //     return view('owner.home');
    // });
    Route::get('/owner', [HomeController::class, 'index'])->name('owner.index');
    Route::post('/owner/filter', [HomeController::class, 'filter'])->name('owner.filter');

    Route::get('/odatadokter', [DataDokterController::class, 'index'])-
>name('odatadokter.index');

    // Routes untuk Jadwal DataDokter
    Route::get('/odatadokter/jadwal/{id}', [JadwalController::class, 'index'])-
>name('odatadokter.jadwal');

    Route::get('/odatapasien', [PasienController::class, 'index'])->name('odatapasien.index');
    Route::get('/datalayanan', [PerawatanController::class, 'index'])->name('layanan.index');
    Route::get('/datalayanan/add', [PerawatanController::class, 'add'])->name('layanan.add');
    Route::post('/datalayanan/insert', [PerawatanController::class, 'insert'])-
>name('layanan.insert');
    Route::get('/datalayanan/edit/{id}', [PerawatanController::class, 'edit'])-
>name('layanan.edit');

```



```

Route::post('/datelayanan/update/{id}', [PerawatanController::class, 'update'])-
>name('layanan.update');
Route::delete('/datelayanan/destroy/{id}', [PerawatanController::class, 'destroy'])-
>name('layanan.destroy');

Route::get('/datapengeluaran', [PengeluaranController::class, 'index'])-
>name('pengeluaran.index');
Route::get('/datapengeluaran/add', [PengeluaranController::class, 'add'])-
>name('pengeluaran.add');
Route::post('/datapengeluaran/insert', [PengeluaranController::class, 'insert'])-
>name('pengeluaran.insert');
Route::get('/datapengeluaran/edit/{id}', [PengeluaranController::class, 'edit'])-
>name('pengeluaran.edit');
Route::post('/datapengeluaran/update/{id}', [PengeluaranController::class, 'update'])-
>name('pengeluaran.update');
Route::delete('/datapengeluaran/destroy/{id}', [PengeluaranController::class, 'destroy'])-
>name('pengeluaran.destroy');

Route::get('/datapemasukan', [PemasukanController::class, 'index'])-
>name('pemasukan.index');

Route::get('/pengeluaran/edit', function () {
    return view('owner.datapengeluaran_edit');
});
Route::get('/pengeluaran/add', function () {
    return view('owner.datapengeluaran_add');
});
Route::get('/odatapasien/rekam/{id}', [RekamController::class, 'index'])-
>name('owner.rekam pasien');
Route::get('/owner/izin', [IzinController::class, 'index'])->name('owner.izin.index');
Route::post('/owner/izin/terima/{id}', [IzinController::class, 'terima'])-
>name('owner.izin.terima');
Route::post('/owner/izin/tolak/{id}', [IzinController::class, 'tolak'])-
>name('owner.izin.tolak');

});

```

## Kode Controller Admin

### 1. DataDokterController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use App\Models\Lokasi;
use App\Models\DataDokter;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use RealRashid\SweetAlert\Facades\Alert;

class DataDokterController extends Controller
{
    protected $datadokter;

    public function __construct(DataDokter $datadokter)
    {
        $this->datadokter = $datadokter;
    }

    public function index()
    {
        $datadokter = [
            'datadokter' => $this->datadokter->allData(),
        ];
        $user = Auth::user();
        if ($user->role === 'owner') {
            return view('owner.datadokter', $datadokter);
        }
    }
}

```

```

        else{

            return view('admin.datadokter', $datadokter);
        }
    }

    public function add()
    {
        $lokasi = Lokasi::all();
        return view('admin.datadokter_add', compact('lokasi'));
    }

    public function insert(Request $request)
    {
        $request->validate([
            'nama' => 'required',
            'alamat' => 'required',
            'nomor_hp' => 'required',
            'lokasi_id' => 'required',
        ],[
            'nama.required' => 'Nama harus diisi!',
            'alamat.required' => 'alamat harus diisi!',
            'nomor_hp.required' => 'nomor_hp harus diisi!',
            'lokasi_id.required' => 'lokasi praktek harus diisi'
        ]);

        $data = [
            'nama' => $request->nama,
            'alamat' => $request->alamat,
            'nomor_hp' => $request->nomor_hp,
            'lokasi_id' => $request->lokasi_id
        ];

        $this->datadokter->addData($data);
        Alert::success('Berhasil!', 'Data dokter berhasil ditambahkan!');
        return redirect('/datadokter');
    }

    public function destroy($dokter_id)
    {
        DB::table('dokter')->where('dokter_id', $dokter_id)->delete();

        // Menampilkan alert sukses dan mengarahkan kembali ke halaman data dokter
        Alert::success('Berhasil!', 'Data dokter berhasil dihapus!');
        return redirect('/datadokter');
    }

    public function edit($id)
    {
        $datadokter = DB::table('dokter')->where('dokter_id', $id)->first();
        return view('admin.datadokter_edit', ['datadokter' => $datadokter]);
    }

    public function detail($id)
    {
        $datajadwal = DB::table('detail_jadwal')->where('dokter_id', $id)->get();
        return view('admin.datadokterjadwal', ['datajadwal' => $datajadwal]);
    }

    public function update(Request $request, $id)
    {
        $request->validate([
            'nama' => 'required',
            'alamat' => 'required',
            'nomor_hp' => 'required',
        ]);

        $data = [
            'nama' => $request->nama,
            'alamat' => $request->alamat,
            'nomor_hp' => $request->nomor_hp,
        ];

        DB::table('dokter')->where('dokter_id', $id)->update($data);
        Alert::success('Berhasil!', 'Data dokter berhasil diupdate!');
        return redirect('/datadokter');
    }
}

```

## 2. HomeController.php

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;
use App\Models\Izin;

class HomeController extends Controller
{
    public function index()
    {
        $today = Carbon::today()->toDateString();
        $absences = Izin::where('tanggal_awal', '<=', $today)
            ->where('tanggal_akhir', '>=', $today)
            ->get();

        $currentMonth = Carbon::now()->format('Y-m');
        $visitsToday = DB::table('reservasi_rekam_medik')
            ->whereDate('tanggal', Carbon::today())
            ->count();
        $visitsThisMonth = $this->getMonthlyVisits($currentMonth);

        $visitStats = [
            'today' => $visitsToday,
            'week1' => $visitsThisMonth['week1'],
            'week2' => $visitsThisMonth['week2'],
            'week3' => $visitsThisMonth['week3'],
            'week4' => $visitsThisMonth['week4'],
            'this_month' => $visitsThisMonth['this_month'],
        ];
        $visits = DB::table('reservasi_rekam_medik')
            ->whereYear('tanggal', Carbon::now()->year)
            ->whereMonth('tanggal', Carbon::now()->month)
            ->count();
        $doctor = DB::table("Dokter")
            ->count();

        $location = DB::table("lokasi")
            ->count();

        // Calculate patient stats
        $patientStats = DB::table('Pasien')
            ->join('reservasi_rekam_medik', 'Pasien.pasien_id', '=',
'reservasi_rekam_medik.pasien_id')
            ->whereYear('reservasi_rekam_medik.tanggal', Carbon::now()->year)
            ->whereMonth('reservasi_rekam_medik.tanggal', Carbon::now()->month)
            ->select(
                DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 < 2 THEN 1 ELSE
0 END) as bayi"),
                DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 2 AND
12 THEN 1 ELSE 0 END) as anak"),
                DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 13 AND
19 THEN 1 ELSE 0 END) as remaja"),
                DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 20 AND
59 THEN 1 ELSE 0 END) as dewasa"),
                DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 >= 60 THEN 1
ELSE 0 END) as lansia")
            )->first();

        // Fetch treatments for the current month
        $treatments = DB::table('Perawatan')
            ->join('perawatan_reservasi', 'Perawatan.perawatan_id', '=',
'perawatan_reservasi.perawatan_id')
            ->join('reservasi_rekam_medik', 'perawatan_reservasi.reservasi_id', '=',
'reservasi_rekam_medik.reservasi_id')
            ->whereYear('reservasi_rekam_medik.tanggal', Carbon::now()->year)
            ->whereMonth('reservasi_rekam_medik.tanggal', Carbon::now()->month)
            ->select('Perawatan.jenis_Perawatan', DB::raw('count(*) as total'))
            ->groupBy('Perawatan.jenis_Perawatan')
            ->get();

        // Fetch doctor performance for the current month
        $doctorPerformance = DB::table('Dokter')
```

```

->join('reservasi_rekam_medik', 'Dokter.dokter_id', '=',
'reservasi_rekam_medik.dokter_id')
->whereYear('reservasi_rekam_medik.tanggal', Carbon::now()->year)
->whereMonth('reservasi_rekam_medik.tanggal', Carbon::now()->month)
->select('Dokter.nama', DB::raw('count(*) as total'))
->groupBy('Dokter.nama')
->get();

$totalIncome = DB::table('Perawatan')
->join('perawatan_reservasi', 'Perawatan.perawatan_id', '=',
'perawatan_reservasi.perawatan_id')
->join('reservasi_rekam_medik', 'perawatan_reservasi.reservasi_id', '=',
'reservasi_rekam_medik.reservasi_id')
->where('reservasi_rekam_medik.draft', 0)
->whereYear('reservasi_rekam_medik.tanggal', Carbon::now()->year)
->whereMonth('reservasi_rekam_medik.tanggal', Carbon::now()->month)
->sum('perawatan_reservasi.harga');

$totalExpenses = DB::table('pengeluaran')
->whereYear('tanggal', Carbon::now()->year)
->whereMonth('tanggal', Carbon::now()->month)
->sum('jumlah_pengeluaran');

// Calculate profit
$profit = $totalIncome - $totalExpenses;

// Determine the user role and return the appropriate view
$user = Auth::user();
if ($user->role === 'owner') {
    return view('owner.home', compact('visitStats', 'currentMonth', 'visits', 'doctor',
'location', 'treatments', 'patientStats', 'doctorPerformance', 'profit'));
}

else{
    return view('admin.home', compact('absences', 'visitStats', 'currentMonth',
'visits', 'doctor', 'location', 'treatments', 'patientStats', 'doctorPerformance', 'profit'));
}

}

public function filter(Request $request)
{
    $currentMonth = Carbon::now()->format('Y-m');
    $month = $request->input('month', $currentMonth);

    $today = Carbon::today()->toDateString();
    $absences = Izin::where('tanggal_awal', '<=', $today)
->where('tanggal_akhir', '>=', $today)
->get();
    $year = substr($month, 0, 4);
    $monthNumber = substr($month, 5, 2);

    $visits = DB::table('reservasi_rekam_medik')
->whereYear('tanggal', $year)
->whereMonth('tanggal', $monthNumber)
->count();
    $doctor = DB::table("Dokter")->count();
    $location = DB::table("lokasi")->count();

    $visitsToday = DB::table('reservasi_rekam_medik')
->whereDate('tanggal', Carbon::today())
->count();

    $visitsInMonth = DB::table('reservasi_rekam_medik')
->whereYear('tanggal', $year)
->whereMonth('tanggal', $monthNumber)
->get();

    $week1 = $visitsInMonth->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 1;
    })->count();

    $week2 = $visitsInMonth->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 2;
    })->count();

    $week3 = $visitsInMonth->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 3;
    })->count();
}

```

```

$week4 = $visitsInMonth->filter(function ($visit) {
    return Carbon::parse($visit->tanggal)->weekOfMonth === 4;
})->count();

$visitStats = [
    'today' => $visitsToday,
    'week1' => $week1,
    'week2' => $week2,
    'week3' => $week3,
    'week4' => $week4,
    'this_month' => $week1+$week2+$week3+$week4,
];

$totalIncome = DB::table('Perawatan')
->join('perawatan_reservasi', 'Perawatan.perawatan_id', '=',
'perawatan_reservasi.perawatan_id')
->join('reservasi_rekam_medik', 'perawatan_reservasi.reservasi_id', '=',
'reservasi_rekam_medik.reservasi_id')
->whereYear('reservasi_rekam_medik.tanggal', $year)
->whereMonth('reservasi_rekam_medik.tanggal', $monthNumber)
->sum('perawatan_reservasi.harga');

// Calculate total expenses for the selected month
$totalExpenses = DB::table('pengeluaran')
->whereYear('tanggal', $year)
->whereMonth('tanggal', $monthNumber)
->sum('jumlah_pengeluaran');

// Calculate profit
$profit = $totalIncome - $totalExpenses;

// Fetch treatments for the selected month
$treatments = DB::table('Perawatan')
->join('perawatan_reservasi', 'Perawatan.perawatan_id', '=',
'perawatan_reservasi.perawatan_id')
->join('reservasi_rekam_medik', 'perawatan_reservasi.reservasi_id', '=',
'reservasi_rekam_medik.reservasi_id')
->whereYear('reservasi_rekam_medik.tanggal', $year)
->whereMonth('reservasi_rekam_medik.tanggal', $monthNumber)
->select('Perawatan.jenis_Perawatan', DB::raw('count(*) as total'))
->groupBy('Perawatan.jenis_Perawatan')
->get();

// Calculate patient stats for the selected month
$patientStats = DB::table('Pasien')

->join('reservasi_rekam_medik', 'Pasien.pasien_id', '=',
'reservasi_rekam_medik.pasien_id')
->whereYear('reservasi_rekam_medik.tanggal', $year)
->whereMonth('reservasi_rekam_medik.tanggal', $monthNumber)
->select(
    DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 < 2 THEN 1 ELSE
0 END) as bayi"),
    DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 2 AND
12 THEN 1 ELSE 0 END) as anak"),
    DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 13 AND
19 THEN 1 ELSE 0 END) as remaja"),
    DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 BETWEEN 20 AND
59 THEN 1 ELSE 0 END) as dewasa"),
    DB::raw("SUM(CASE WHEN DATEDIFF(CURDATE(), tanggal_lahir) / 365.25 >= 60 THEN 1
ELSE 0 END) as lansia")
)->first();

// Fetch doctor performance for the selected month
$doctorPerformance = DB::table('Dokter')
->join('reservasi_rekam_medik', 'Dokter.dokter_id', '=',
'reservasi_rekam_medik.dokter_id')
->whereYear('reservasi_rekam_medik.tanggal', $year)
->whereMonth('reservasi_rekam_medik.tanggal', $monthNumber)
->select('Dokter.nama', DB::raw('count(*) as total'))
->groupBy('Dokter.nama')
->get();

// Determine the user role and return the appropriate view
$user = Auth::user();
if ($user->role === 'owner') {
    return view('owner.home', compact('absences', 'visitStats', 'currentMonth',
'month', 'visits', 'doctor', 'location', 'treatments', 'patientStats', 'doctorPerformance',
'profit'));
} else {

```

```

        return view('admin.home', compact('absences', 'visitStats', 'currentMonth',
'month', 'visits', 'doctor', 'location', 'treatments', 'patientStats', 'doctorPerformance',
'profit'));
    }
}

private function getMonthlyVisits($month)
{
    $year = substr($month, 0, 4);
    $monthNumber = substr($month, 5, 2);

    $visits = DB::table('reservasi_rekam_medik')
->whereYear('tanggal', $year)
->whereMonth('tanggal', $monthNumber)
->get();

    $week1 = $visits->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 1;
    })->count();

    $week2 = $visits->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 2;
    })->count();

    $week3 = $visits->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 3;
    })->count();

    $week4 = $visits->filter(function ($visit) {
        return Carbon::parse($visit->tanggal)->weekOfMonth === 4;
    })->count();

    $thisMonth = $visits->count();

    return [
        'week1' => $week1,
        'week2' => $week2,
        'week3' => $week3,
        'week4' => $week4,
        'this_month' => $thisMonth,
    ];
}
}

```

### 3. JadwalController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use App\Models\Dokter;
use App\Models\DetailJadwal;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use RealRashid\SweetAlert\Facades\Alert;

class JadwalController extends Controller
{
    protected $datadokter;

    public function __construct(DetailJadwal $datadokter)
    {
        $this->datadokter = $datadokter;
    }

    public function index($id)
    {
        $dokter = Dokter::where("dokter_id", $id)->first();
        $dokterjadwal = DetailJadwal::with(['dokter'])->where("dokter_id", $id)->first();
        $datajadwal = DB::table('detail_jadwal')->where('dokter_id', $id)->get();

        $user = Auth::user();
        if ($user->role === 'owner') {
            return view('owner.datadokterjadwal', ['datajadwal' => $datajadwal, 'datadokter'
=> $dokter]);
        }
    }
}

```

```

        return view('admin.datadokterjadwal', ['datajadwal' => $datajadwal, 'datadokter' =>
$ddokter]);
        // $datadokter = [
        //     'datadokter' => $this->datadokter->allData(),
        // ];

        // return view('admin.datadokter', $datadokter);
    }

    public function add($id)
    {
        return view('admin.datadokterjadwal_add', ['id' => $id]);
    }

    public function insert(Request $request, $id)
    {
        // Validasi input form
        $request->validate([
            'hari' => 'required',
            'sesi' => 'required',
        ], [
            'hari.required' => 'Hari harus diisi!',
            'sesi.required' => 'Sesi harus diisi!',
        ]);

        // Menyiapkan data yang akan disimpan
        $data = [
            'hari' => $request->hari,
            'sesi' => $request->sesi,
            'dokter_id' => $id,
        ];

        // Menyimpan data menggunakan model (asumsi ada method addData di model yang
digunakan)
        $this->datadokter->addData($data);

        // Menampilkan pesan sukses menggunakan SweetAlert (asumsi Alert diimport dengan
benar)
        Alert::success('Berhasil!', 'Data jadwal berhasil ditambahkan!');

        // Redirect ke halaman yang diinginkan dengan sintaks kurung keriting ganda
        return redirect("/datadokter/jadwal/{$id}");
    }

    public function destroy($jadwal_id)
    {
        // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
        DB::table('detail_jadwal')->where('jadwal_id', $jadwal_id)->delete();

        // Menampilkan alert sukses dan mengarahkan kembali ke halaman data dokter
        Alert::success('Berhasil!', 'Data jadwal dokter berhasil dihapus!');
        $id_dokter = DB::table('detail_jadwal')->where('jadwal_id', $jadwal_id)-
>value('dokter_id');
        if($id_dokter==NULL){
            return redirect("/datadokter");
        }
        else{
            return redirect("/datadokter/jadwal/{$id_dokter}");
        }
    }

    public function edit($id)
    {
        $datadokter = DB::table('detail_jadwal')->where('jadwal_id', $id)->first();
        return view('admin.datadokterjadwal_edit', ['datadokter' => $datadokter]);
    }

    public function update(Request $request, $id)
    {
        $request->validate([
            'hari' => 'required',
            'sesi' => 'required',
        ], [
            'hari.required' => 'Hari harus diisi!',
            'sesi.required' => 'Sesi harus diisi!',
        ]);

        $data = [
            'hari' => $request->hari,

```

```

        'sesi' => $request->sesi,
    ];

    DB::table('detail_jadwal')->where('jadwal_id', $id)->update($data);
    $id_dokter = DB::table('detail_jadwal')->where('jadwal_id', $id)->value('dokter_id');
    Alert::success('Berhasil!', 'Data Jadwal berhasil diupdate!');
    return redirect("/datadokter/jadwal/{$id_dokter}");
}
}

```

#### 4. PasienController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use Illuminate\Http\Request;
use App\Models\Pasien;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class PasienController extends Controller
{
    protected $pasien;

    public function __construct(Pasien $pasien)
    {
        $this->pasien = $pasien;
    }

    public function index()
    {
        $reservasi = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('draft', 0)
        ->select('reservasi_rekam_medik.*',
        'lokasi.*',
        "pasien.*",
        'dokter.nama as nama_dokter')->get();

        $user = Auth::user();
        if ($user->role === 'owner') {
            return view('owner.datapasien', compact('reservasi'));
        } elseif($user->role === 'dokter' || $user->role === 'Dokter') {
            return view('dokter.datapasien', compact('reservasi'));
        }
        else{
            return view('admin.datapasien', compact('reservasi'));
        }
    }

    public function add()
    {
        $user = Auth::user();
        if($user->role === 'dokter' || $user->role === 'Dokter') {
            return view('dokter.datapasien_add');
        }
        else{
            return view('admin.datapasien_add');
        }
    }

    public function insert(Request $request)
    {
        $request->validate([
            'nama' => 'required',
            'tempat_lahir' => 'required',
            'tanggal_lahir' => 'required',
            'pekerjaan' => 'required',
            'alamat' => 'required',
            'no_Telp' => 'required',
        ],[

```



```

        'nama.required' => 'Nama harus diisi!',
        'tempat_lahir.required' => 'tempat_lahir harus diisi!',
        'tanggal_lahir.required' => 'tanggal_lahir harus diisi!',
        'pekerjaan.required' => 'pekerjaan harus diisi!',
        'alamat.required' => 'alamat harus diisi!',
        'no_Telp.required' => 'no_Telp harus diisi!',

    ]);

    $data = [
        'nama' => $request->nama,
        'tempat_lahir' => $request->tempat_lahir,
        'tanggal_lahir' => $request->tanggal_lahir,
        'pekerjaan' => $request->pekerjaan,
        'alamat' => $request->alamat,
        'no_Telp' => $request->no_Telp,

    ];

    $this->pasien->addData($data);
    Alert::success('Berhasil!', 'Data pasien berhasil ditambahkan!');
    $user = Auth::user();
    if($user->role === 'dokter' || $user->role === 'Dokter') {
        return redirect('/ddatapasien');
    }
    else{
        return redirect('/adatapasien');
    }
}

public function destroy($pasien_id)
{
    $reservasi = DB::table('reservasi_rekam_medik')
    ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
    ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
    ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
    ->where('draft', 0)
    ->select('reservasi_rekam_medik.*',
        'lokasi.*',
        "pasien.*",
        'dokter.nama as nama_dokter')->get();

    // Setelah itu, hapus entri dari tabel pasien
    DB::table('pasien')->where('pasien_id', $pasien_id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data pasien
    Alert::success('Berhasil!', 'Data pasien berhasil dihapus!');
    $user = Auth::user();
    if($user->role === 'dokter' || $user->role === 'Dokter') {
        return redirect('/ddatapasien');
    }
    else{
        return redirect('/adatapasien');
    }
}

public function edit($id)
{
    $pasien = DB::table('pasien')->where('pasien_id', $id)->first();

    $user = Auth::user();
    if($user->role === 'dokter' || $user->role === 'Dokter') {
        return view('dokter.datapasien_edit', ['pasien' => $pasien]);
    }
    else{
        return view('admin.datapasien_edit', ['pasien' => $pasien]);
    }
}

public function update(Request $request, $id)
{
    $request->validate([
        'nama' => 'required',
        'tempat_lahir' => 'required',
        'tanggal_lahir' => 'required',
        'pekerjaan' => 'required',
        'alamat' => 'required',
        'no_Telp' => 'required',
    ]);

```

```

        $data = [
            'nama' => $request->nama,
            'tempat_lahir' => $request->tempat_lahir,
            'tanggal_lahir' => $request->tanggal_lahir,
            'pekerjaan' => $request->pekerjaan,
            'alamat' => $request->alamat,
            'no_Telp' => $request->no_Telp,
        ];

        DB::table('pasien')->where('pasien_id', $id)->update($data);
        Alert::success('Berhasil!', 'Data pasien berhasil diupdate!');
        $user = Auth::user();
        if($user->role === 'dokter' || $user->role === 'Dokter') {
            return redirect("/ddatapasien");
        }
        else{
            return redirect('/adatapasien');
        }
    }
}

```

## 5. RekamController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use App\Models\RekamMedik;
use App\Models\Reservasi;
use App\Models\PerawatanReservasi;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use App\Models\Perawatan;

class RekamController extends Controller
{
    protected $rekam_medik;

    public function __construct(RekamMedik $rekam_medik)
    {
        $this->rekam_medik = $rekam_medik;
    }

    public function index($id)
    {
        // Cari semua pasien terkait dengan dokter_id melalui rekam_medik
        $reservasiIds = DB::table('reservasi_rekam_medik')->where('reservasi_id', $id)->pluck('pasien_id');

        $reservasi_dokter = DB::table("reservasi_rekam_medik")
            ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
            ->where("reservasi_id", $id)
            ->first();
        $reservasi_pasien = DB::table("reservasi_rekam_medik")
            ->join("pasien", "reservasi_rekam_medik.pasien_id", "=", "pasien.pasien_id")
            ->where("reservasi_id", $id)
            ->first();

        $perawatan_reservasi = PerawatanReservasi::with(['reservasi', 'perawatan'])
            ->where('reservasi_id', $id)
            ->get();

        // $rekam_medik = DB::table('rekam_medik')
        // ->whereIn('rekam_medik.reservasi_id', $reservasiIds)
        // ->first();

        // $rekam_medik_perawatan = DB::table('rekam_medik')
        // ->join('perawatan', 'rekam_medik.perawatan_id', '=', 'perawatan.perawatan_id')
        // ->whereIn('rekam_medik.reservasi_id', $reservasiIds)
        // ->first();

        $reservasi_rekam_medik = Reservasi::with('perawatan')->findOrFail($id);
        $user = Auth::user();
        if ($user->role === 'owner') {
            return view('owner.rekammedik', compact('reservasi_dokter', 'reservasi_pasien', 'reservasi_rekam_medik', 'perawatan_reservasi'));
        }
    }
}

```

```

    }
    elseif($user->role === 'dokter' || $user->role === 'Dokter') {
        return view('dokter.rekammedik', compact('reservasi_dokter', 'reservasi_pasien'
, 'reservasi_rekam_medik', 'perawatan_reservasi'));
    }
    else{
        return view('admin.rekammedik', compact('reservasi_dokter', 'reservasi_pasien'
, 'reservasi_rekam_medik', 'perawatan_reservasi'));
    }
    // $rekam_medik = [
    //     'rekam_medik' => $this->rekam_medik->allData(),
    // ];

    // return view('admin.rekam_medik', $rekam_medik);
}

public function add($id)
{
    $user = Auth::user();
    if($user->role === 'dokter' || $user->role === 'Dokter') {
        return view('dokter.rekam_medik_add', ['id' => $id]);
    }
    else
    {
        return view('admin.rekam_medik_add', ['id' => $id]);
    }
}

public function edit($id)
{
    $perawatan = DB::table('perawatan')->get();

    $reservasi = DB::table('reservasi_rekam_medik')->where('reservasi_id', $id)->first();
    $user = Auth::user();

    if($user->role === 'dokter' || $user->role === 'Dokter') {
        return view('dokter.rekammedik_edit', ['reservasi' => $reservasi],
compact('perawatan'));
    }
    else
    {
        return view('admin.rekammedik_edit', ['reservasi' => $reservasi],
compact('perawatan'));
    }
}

public function update(Request $request, $id)
{
    // Validasi data yang dikirimkan dari formulir
    $validatedData = $request->validate([
        'golongan_darah' => 'required',
        'tekanan_darah' => 'required',
        'penyakit_jantung' => 'required',
        'diabetes' => 'required',
        'hepatitis' => 'required',
        'penyakit_lainnya' => 'required',
        'alergi_makanan' => 'required',
        'alergi_obat' => 'required',
        'keluhan' => 'required',
        'perawatan_id' => 'required',
        'gigi' => 'required',
        'perawatan_id' => 'required|array|max:2',
        'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
    ]);

    // Perbarui data rekam medik dengan data yang dikirimkan dari formulir

    // Mengambil data harga dan estimasi waktu dari tabel perawatan berdasarkan ID

    $syncData = [];
    foreach ($validatedData['perawatan_id'] as $perawatanId) {
        $perawatan = Perawatan::find($perawatanId);
        $syncData[$perawatanId] = [
            'harga' => $perawatan->harga,
            'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
        ];
    }
}

```

```

        $reservasi = Reservasi::findOrFail($id);
        $reservasi->update($validatedData);
        $reservasi->perawatan()->sync($syncData);

        Alert::success('Berhasil!', 'Data dokter berhasil diupdate!');
        $user = Auth::user();
        if($user->role === 'dokter' || $user->role === 'Dokter') {
            return redirect()->route("dokter.rekamkasien",[$id]);
        }
        else
        {
            return redirect()->route("admin.rekamkasien",[$id]);
        }
    }
}

```

## 6. ReservasiController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use App\Models\Pasien;
use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Lokasi;
use App\Models\Perawatan;
use App\Models\Reservasi;
use App\Models\RekamMedik;
use App\Models\Dokter;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use App\Models\DetailJadwal;
use Illuminate\Support\Carbon;
use App\Services\WhatsAppService;
use Exception;
use Illuminate\Support\Facades\Log;
use Illuminate\Validation\ValidationException;
use DateTime;
use DateInterval;

class ReservasiController extends Controller
{
    protected $user;
    protected $whatsAppService;

    public function __construct(User $user, WhatsAppService $whatsAppService)
    {
        $this->user = $user;
        $this->whatsAppService = $whatsAppService;
    }

    public function index()
    {
        $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join('dokter', "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('draft', 0)
        ->select('reservasi_rekam_medik.*',
            'lokasi.*',
            "pasien.*",
            'dokter.nama as nama_dokter')
        ->orderBy('tanggal', 'desc')
        ->get();

        $reservasi_admin = DB::table("reservasi_rekam_medik")
        ->join("pasien", "reservasi_rekam_medik.pasien_id", "=", "pasien.pasien_id")->get();

        return view('admin.reservasiadmin', compact('reservasi_rekam_medik'));
    }
}

```

```

public function pasien()
{
    $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
->where('draft', NULL)
->select('reservasi_rekam_medik.*',
'lokasi.*',
"pasien.*",
'dokter.nama as nama_dokter')->get();

    return view('admin.reservasipasien', compact('reservasi_rekam_medik'));
}

public function add()
{
    $lokasi = Lokasi::all();
    $perawatan = Perawatan::all();
    $dokter = Dokter::all();

    $tomorrow = Carbon::tomorrow()->format('Y-m-d');

    // Generate available timeslots
    $timeslots = [];
    for ($hour = 10; $hour < 21; $hour++) {
        for ($minute = 0; $minute < 60; $minute += 30) {
            $start = Carbon::createFromTime($hour, $minute);
            $end = $start->copy()->addMinutes(30);
            $timeslots[] = [
                'start' => $start->format('H:i'),
                'end' => $end->format('H:i')
            ];
        }
    }

    $user = Auth::user();
    if ($user->role === 'Pasien') {
        return view('pasien.reservasipasien_add', compact('lokasi', 'dokter', 'perawatan',
'tomorrow', 'timeslots'));
    } elseif ($user->role === 'Admin' || $user->role === 'admin') {
        return view('admin.reservasiadmin_add', compact('lokasi', 'dokter', 'perawatan',
'tomorrow', 'timeslots'));
    }
}

public function insert(Request $request)
{
    try {
        $validatedData = $request->validate([
            'nama' => 'required|string|max:255',
            'tempat_lahir' => 'required|string|max:255',
            'tanggal_lahir' => 'required|date',
            'pekerjaan' => 'required|string|max:255',
            'alamat' => 'required|string',
            'no_Telp' => 'required|string|max:15',
            'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
            'dokter_id' => 'required|integer',
            'tanggal' => 'required|date',
            'jam_mulai' => 'required',
            'perawatan_id' => 'required|array|max:2',
            'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
        ]);

        $tanggal = Carbon::parse($validatedData['tanggal'])->format('Y-m-d');

        // Calculate total estimated time
        $totalEstimasi = 0;
        foreach ($validatedData['perawatan_id'] as $perawatanId) {
            $perawatan = Perawatan::find($perawatanId);
            $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
        }

        $startTime = new DateTime($validatedData['jam_mulai']);
        $endTime = (clone $startTime)->modify("+{$totalEstimasi} minutes")->format('H:i');

        // Check for time conflicts
    }
}

```

```

        $conflictingReservation = Reservasi::where('tanggal', $tanggal)
        ->where(function ($query) use ($validatedData, $endTime) {
            $query->whereBetween('jam_mulai', [$validatedData['jam_mulai'], $endTime])
            ->orWhereBetween('jam_selesai', [$validatedData['jam_mulai'],
$endTime])
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai',
[$validatedData['jam_mulai']])
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai', [$endTime]);
        })
        ->exists();

    // Create new reservation

    $pasien = Pasien::create([
        'nama' => $validatedData['nama'],
        'tempat_lahir' => $validatedData['tempat_lahir'],
        'tanggal_lahir' => $validatedData['tanggal_lahir'],
        'pekerjaan' => $validatedData['pekerjaan'],
        'alamat' => $validatedData['alamat'],
        'no_Telp' => $validatedData['no_Telp'],
    ]);

    $reservasi_rekam_medik = Reservasi::create([
        'pasien_id' => $pasien['pasien_id'],
        'lokasi_id' => $validatedData['lokasi_id'],
        'dokter_id' => $validatedData['dokter_id'],
        'tanggal' => $tanggal,
        'jam_mulai' => $validatedData['jam_mulai'],
        'jam_selesai' => $endTime,
        'status_penginput' => 1,
        'draft' => 0
    ]);

    $syncData = [];
    foreach ($validatedData['perawatan_id'] as $perawatanId) {
        $perawatan = Perawatan::find($perawatanId);
        $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
        $syncData[$perawatanId] = [
            'harga' => $perawatan->harga,
            'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
        ];
    }

    $reservasi_rekam_medik->perawatan()->attach($syncData);

    Alert::success('Success', 'Reservasi berhasil ditambahkan!');
    return redirect()->route('reservasi.index');
} catch (Exception $e) {
    dd('simpan data gagal : ', $e);
}
}

public function getBookedTimes(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $tanggal = $request->query('tanggal');
    $tanggalDate = Carbon::parse($tanggal);

    $bookedTimes = Reservasi::where('dokter_id', $dokter_id)
        ->where('tanggal', $tanggalDate)
        ->get(['jam_mulai', 'jam_selesai']);

    return response()->json($bookedTimes);
}

public function getDokterByLokasi(Request $request)
{
    $lokasi_id = $request->query('lokasi_id');
    $dokter = Dokter::where('lokasi_id', $lokasi_id)->get();
    return response()->json($dokter);
}

public function getAvailableDays(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)->pluck('hari');
    return response()->json($jadwals);
}

```

```

}

public function getSesiByDokterAndHari(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $hari = $request->query('hari');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)
        ->where('hari', $hari)
        ->pluck('sesi');
    return response()->json($jadwals);
}

public function destroy($id)
{
    // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
    DB::table('reservasi_rekam_medik')->where('reservasi_id', $id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data user
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik berhasil dihapus!');
    return redirect('/reservasi/admin');
}

public function terima($id)
{
    $reservasi_rekam_medik = Reservasi::where('reservasi_id', $id)->firstOrFail();

    // Prepare data for reservasi_rekam_medik
    $reservasiData = [
        'draft' => 0,
    ];
    // Update the reservasi_rekam_medik
    $reservasi_rekam_medik->update($reservasiData);
    $reservasi = Reservasi::with(['pasien'])
        ->where('reservasi_id', $id)
        ->firstOrFail();

    $this->whatsappService->sendWhatsAppMessage($reservasi->pasien->no_Telp, 'Reservasi
Anda telah diterima, silahkan cek status reservasi anda pada akun website Xenon Dental House
anda');

    // Redirect or return a response
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien diterima !');
    return redirect()->route('reservasi.pasien');
}

public function tolak($id)
{
    $reservasi_rekam_medik = Reservasi::where('reservasi_id', $id)->firstOrFail();

    // Prepare data for reservasi_rekam_medik
    $reservasiData = [
        'draft' => 1,
    ];
    // Update the reservasi_rekam_medik
    $reservasi_rekam_medik->update($reservasiData);

    $reservasi = Reservasi::with(['pasien'])
        ->where('reservasi_id', $id)
        ->firstOrFail();

    $this->whatsappService->sendWhatsAppMessage($reservasi->pasien->no_Telp, 'Reservasi
Anda telah ditolak, silahkan cek status reservasi anda pada akun website Xenon Dental House
anda');

    // Redirect or return a response
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien ditolak !');
    return redirect()->route('reservasi.pasien');
}

public function edit($id)
{
    $lokasi = Lokasi::all();
    $perawatan = Perawatan::all();
    $dokter = Dokter::all();
    $reservasi = Reservasi::with(['pasien', 'dokter', 'lokasi'])

```

```

        ->where('reservasi_id', $id)
        ->first();

$tomorrow = Carbon::tomorrow()->format('Y-m-d');

// Generate available timeslots
$timeslots = [];
for ($hour = 10; $hour < 21; $hour++) {
    for ($minute = 0; $minute < 60; $minute += 30) {
        $start = Carbon::createFromTime($hour, $minute);
        $end = $start->copy()->addMinutes(30);
        $timeslots[] = [
            'start' => $start->format('H:i'),
            'end' => $end->format('H:i')
        ];
    }
}

return view('admin.reservasiadmin_edit', compact('lokasi', 'dokter',
'perawatan', 'reservasi', 'tomorrow'));
}

public function update(Request $request, $reservasi_id)
{
    try {
        $validatedData = $request->validate([
            'nama' => 'required|string|max:255',
            'tempat_lahir' => 'required|string|max:255',
            'tanggal_lahir' => 'required|date',
            'pekerjaan' => 'required|string|max:255',
            'alamat' => 'required|string',
            'no_Telp' => 'required|string|max:15',
            'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
            'dokter_id' => 'required|integer',
            'tanggal' => 'required|date',
            'jam_mulai' => 'required',
            'perawatan_id' => 'required|array|max:2',
            'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
        ]);

        $tanggal = Carbon::parse($validatedData['tanggal'])->format('Y-m-d');

        // Calculate total estimated time
        $totalEstimasi = 0;
        foreach ($validatedData['perawatan_id'] as $perawatanId) {
            $perawatan = Perawatan::find($perawatanId);
            $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
        }

        $startTime = new DateTime($validatedData['jam_mulai']);
        $endTime = (clone $startTime)->modify("+{$totalEstimasi} minutes")->format('H:i');

        // Check for time conflicts
        $conflictingReservation = Reservasi::where('tanggal', $tanggal)
        ->where('reservasi_id', '!=', $reservasi_id)
        ->where(function ($query) use ($validatedData, $endTime) {
            $query->whereBetween('jam_mulai', [$validatedData['jam_mulai'], $endTime])
            ->orWhereBetween('jam_selesai', [$validatedData['jam_mulai'],
$endTime])
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai',
[$validatedData['jam_mulai']])
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai', [$endTime]);
        })
        ->exists();

        // if ($conflictingReservation) {
        //     Alert::error('Error', 'Waktu yang dipilih bentrok dengan reservasi lain!');
        //     return redirect()->back()->withInput();
        // }

        // Update existing reservation
        $reservasi = Reservasi::findOrFail($reservasi_id);
        $pasien = Pasien::findOrFail($reservasi->pasien_id);

        $pasien->update([
            'nama' => $validatedData['nama'],
            'tempat_lahir' => $validatedData['tempat_lahir'],
            'tanggal_lahir' => $validatedData['tanggal_lahir'],
            'pekerjaan' => $validatedData['pekerjaan'],

```



```

        'alamat' => $validatedData['alamat'],
        'no_Telp' => $validatedData['no_Telp'],
    ]));

    $reservasi->update([
        'lokasi_id' => $validatedData['lokasi_id'],
        'dokter_id' => $validatedData['dokter_id'],
        'tanggal' => $tanggal,
        'jam_mulai' => $validatedData['jam_mulai'],
        'jam_selesai' => $endTime,
        'status_penginput' => 1,
        'draft' => 0
    ]);

    $syncData = [];
    foreach ($validatedData['perawatan_id'] as $perawatanId) {
        $perawatan = Perawatan::find($perawatanId);
        $syncData[$perawatanId] = [
            'harga' => $perawatan->harga,
            'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
        ];
    }

    $reservasi->perawatan()->sync($syncData);

    Alert::success('Success', 'Reservasi berhasil diperbarui!');
    return redirect()->route('reservasi.index');
} catch (Exception $e) {
    dd('update data gagal : ', $e);
}
}
}

```

## 7. UserController.php

```

<?php

namespace App\Http\Controllers\Admin;

use Session;
use App\Models\User;
use App\Models\Dokter;
use App\Models\Lokasi;
use App\Models\DataDokter;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Hash;
use RealRashid\SweetAlert\Facades\Alert;

class UserController extends Controller
{
    protected $user;

    public function __construct(User $user)
    {
        $this->user = $user;
    }

    public function index()
    {
        $user = [
            'user' => $this->user->allData(),
        ];

        return view('admin.datauser', $user);
    }

    public function add()
    {
        return view('admin.datauser_add');
    }

    public function insert(Request $request)
    {
        $request->validate([
            'name' => 'required',
            'email' => 'required',
            'password' => 'required',
            'role' => 'required',

```

```

],[
    'name.required' => 'Nama harus diisi!',
    'email.required' => 'email harus diisi!',
    'password.required' => 'role harus diisi!',
    'role.required' => 'role harus diisi!',
]);

$;

$data = [
    'name' => $request->name,
    'email' => $request->email,
    'password' => Hash::make($request->password),
    'role' => $request->role,
];

$this->user->addData($data);
Alert::success('Berhasil!', 'Data user berhasil ditambahkan!');
return redirect('/datauser');
}

public function destroy($id)
{
    // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
    DB::table('users')->where('id', $id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data user
    Alert::success('Berhasil!', 'Data user berhasil dihapus!');
    return redirect('/datauser');
}

public function add_dokter()
{
    $lokasi = Lokasi::all();
    return view('admin.datauserdokter_add', compact('lokasi'));
}

public function insert_dokter(Request $request)
{
    $request->validate([
        'nama' => 'required',
        'alamat' => 'required',
        'nomor_hp' => 'required',
        'lokasi_id' => 'required',
        'email' => 'required',
        'password' => 'required',
    ],[
        'nama.required' => 'Nama harus diisi!',
        'alamat.required' => 'alamat harus diisi!',
        'nomor_hp.required' => 'nomor_hp harus diisi!',
        'lokasi_id.required' => 'lokasi praktek harus diisi!',
        'email.required' => 'email harus diisi!',
        'password.required' => 'role harus diisi!',
    ]);

    $data = [
        'name' => $request->nama,
        'email' => $request->email,
        'password' => Hash::make($request->password),
        'role' => 'Dokter',
    ];

    $id_user = $this->user->addData($data);

    Dokter::create([
        'nama' => $request->nama,
        'alamat' => $request->alamat,
        'nomor_hp' => $request->nomor_hp,
        'lokasi_id' => $request->lokasi_id,
        'user_id' => $id_user
    ]);

    Alert::success('Berhasil!', 'Data akun dokter berhasil ditambahkan!');
    return redirect('/datauser');
}

```

```

public function edit($id)
{
    $user = DB::table('users')->where('id', $id)->first();
    return view('admin.datauser_edit', ['user' => $user]);
}

public function update(Request $request, $id)
{
    $request->validate([
        'name' => 'required',
        'email' => 'required',
        'password' => 'required',
        'role' => 'required',
    ]);

    $data = [
        'name' => $request->name,
        'email' => $request->email,
        'password' => $request->password,
        'role' => $request->role,
    ];

    DB::table('users')->where('id', $id)->update($data);
    Alert::success('Berhasil!', 'Data user berhasil diupdate!');
    return redirect('/datauser');
}
}

```

## Code Controller Auth

### 1. AuthenticatedSessionController.php

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     */
    public function create(): View
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     */
    public function store(LoginRequest $request)
    {
        $request->authenticate();

        $request->session()->regenerate();

        $user = Auth::user();

        if (!$user) {
            return response()->json(['success' => false, 'message' => 'Authentication
failed.'], 401);
        }

        $role = $user->role;
        $redirectUrl = '/admin'; // Default redirect

        if ($role === 'owner' || $role === 'Owner') {
            return redirect('/owner');
        } elseif ($role === 'dokter' || $role === 'Dokter') {
            return redirect('/dokter');
        } elseif ($role === 'pasien' || $role === 'Pasien') {
            return redirect('/home');
        }
    }
}

```

```

        return redirect("/admin");
    }

    /**
     * Destroy an authenticated session.
     */
    public function destroy(Request $request): RedirectResponse
    {
        Auth::guard('web')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}

```

## 2. EmailVerificationPromptController.php

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Http\Requests\Auth\LoginRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\View\View;

class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     */
    public function create(): View
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     */
    public function store(LoginRequest $request)
    {
        $request->authenticate();

        $request->session()->regenerate();

        $user = Auth::user();

        if (!$user) {
            return response()->json(['success' => false, 'message' => 'Authentication
failed.'], 401);
        }

        $role = $user->role;
        $redirectUrl = '/admin'; // Default redirect

        if ($role === 'owner' || $role === 'Owner') {
            return redirect('/owner');
        } elseif ($role === 'dokter' || $role === 'Dokter') {
            return redirect('/dokter');
        } elseif ($role === 'pasien' || $role === 'Pasien') {
            return redirect('/home');
        }

        return redirect("/admin");
    }

    /**
     * Destroy an authenticated session.
     */
    public function destroy(Request $request): RedirectResponse
    {
        Auth::guard('web')->logout();
    }
}

```

```

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect('/');
    }
}

```

### 3. NewPasswordController.php

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Auth\Events\PasswordReset;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Password;
use Illuminate\Support\Str;
use Illuminate\Validation\Rules;
use Illuminate\View\View;

class NewPasswordController extends Controller
{
    /**
     * Display the password reset view.
     */
    public function create(Request $request): View
    {
        return view('auth.reset-password', ['request' => $request]);
    }

    /**
     * Handle an incoming new password request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'token' => ['required'],
            'email' => ['required', 'email'],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);

        // Here we will attempt to reset the user's password. If it is successful we
        // will update the password on an actual user model and persist it to the
        // database. Otherwise we will parse the error and return the response.
        $status = Password::reset(
            $request->only('email', 'password', 'password_confirmation', 'token'),
            function ($user) use ($request) {
                $user->forceFill([
                    'password' => Hash::make($request->password),
                    'remember_token' => Str::random(60),
                ])->save();

                event(new PasswordReset($user));
            }
        );

        // If the password was successfully reset, we will redirect the user back to
        // the application's home authenticated view. If there is an error we can
        // redirect them back to where they came from with their error message.
        return $status == Password::PASSWORD_RESET
            ? redirect()->route('login')->with('status', __($status))
            : back()->withInput($request->only('email'))
                ->withErrors(['email' => __($status)]);
    }
}

```

### 4. PasswordController.php

```

<?php

namespace App\Http\Controllers\Auth;

```

```

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules\Password;

class PasswordController extends Controller
{
    /**
     * Update the user's password.
     */
    public function update(Request $request): RedirectResponse
    {
        $validated = $request->validateWithBag('updatePassword', [
            'current_password' => ['required', 'current_password'],
            'password' => ['required', Password::defaults(), 'confirmed'],
        ]);

        $request->user()->update([
            'password' => Hash::make($validated['password']),
        ]);

        return back()->with('status', 'password-updated');
    }
}

```

## 5. PasswordResetLinkController.php

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Password;
use Illuminate\View\View;

class PasswordResetLinkController extends Controller
{
    /**
     * Display the password reset link request view.
     */
    public function create(): View
    {
        return view('auth.forgot-password');
    }

    /**
     * Handle an incoming password reset link request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'email' => ['required', 'email'],
        ]);

        // We will send the password reset link to this user. Once we have attempted
        // to send the link, we will examine the response then see the message we
        // need to show to the user. Finally, we'll send out a proper response.
        $status = Password::sendResetLink(
            $request->only('email')
        );

        return $status == Password::RESET_LINK_SENT
            ? back()->with('status', __($status))
            : back()->withInput($request->only('email'))
                ->withErrors(['email' => __($status)]);
    }
}

```

## 6. RegisteredUserController.php

```

<?php

```

```

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
use Illuminate\View\View;

class RegisteredUserController extends Controller
{
    /**
     * Display the registration view.
     */
    public function create(): View
    {
        return view('auth.register');
    }

    /**
     * Handle an incoming registration request.
     *
     * @throws \Illuminate\Validation\ValidationException
     */
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255',
                'unique:'.User::class],
            'password' => ['required', 'confirmed', Rules\Password::defaults()],
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => Hash::make($request->password),
            'role' => 'Pasien',
        ]);

        event(new Registered($user));

        Auth::login($user);

        $user = Auth::user();
        if ($user->role === 'owner' || $user->role === 'Owner' ) {
            return redirect('/owner');
        }
        elseif ($user->role === 'dokter' || $user->role === 'Dokter') {
            return redirect('/dokter');
        }
        elseif ($user->role === 'Pasien' || $user->role === 'pasien') {
            return redirect('/home');
        }

        return redirect("/admin");
    }
}

```

## 7. VerifyEmailController.php

```

<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Illuminate\Auth\Events\Verified;
use Illuminate\Foundation\Auth\EmailVerificationRequest;
use Illuminate\Http\RedirectResponse;

class VerifyEmailController extends Controller
{
    /**
     * Mark the authenticated user's email address as verified.
     */
    public function __invoke(EmailVerificationRequest $request): RedirectResponse
    {

```

```

        if ($request->user()->hasVerifiedEmail()) {
            return redirect()->intended(route('admin', absolute: false).'?verified=1');
        }

        if ($request->user()->markEmailAsVerified()) {
            event(new Verified($request->user()));
        }

        return redirect()->intended(route('admin', absolute: false).'?verified=1');
    }
}

```

## Kode Controller Dokter

### 1. DokterHomeController.php

```

<?php

namespace App\Http\Controllers\Dokter;

use App\Http\Controllers\Controller;
use App\Models\DetailJadwal;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Carbon\Carbon;
use Illuminate\Support\Facades\Auth;
use App\Models\Izin;

class DokterHomeController extends Controller
{
    public function index()
    {
        $user = Auth::user();
        $userid = $user->id;

        // $jadwal = DetailJadwal::with('dokter')
        // ->where('user_id', $userid)
        // ->get();

        $jadwalid = DB::table('detail_jadwal')
        ->join('dokter', 'detail_jadwal.dokter_id', '=', 'dokter.dokter_id')
        ->where('dokter.user_id', $userid)->get();

        // Determine the user role and return the appropriate view

        return view('dokter.home', compact("jadwalid"));
    }
}

```

### 2. IzinController.php

```

<?php

namespace App\Http\Controllers\Dokter;

use Session;
use Exception;
use App\Models\Izin;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;
use RealRashid\SweetAlert\Facades\Alert;

class IzinController extends Controller
{
    protected $izin;

    public function __construct(izin $izin)
    {
        $this->izin = $izin;
    }
}

```



```

public function index()
{
    $izin = [
        'izin' => $this->izin->allData(),
    ];

    $izin_user = Izin::with(['users'])->get();
    $izin_user_owner = Izin::with(['users'])
        ->whereNull('status')
        ->get();

    $user = Auth::user();
    if($user->role === 'Dokter'){
        return view('dokter.dataabsen', compact('izin', 'izin_user'));
    }
    else{
        return view('owner.dataizin', compact('izin', 'izin_user_owner'));
    }
}

public function add()
{
    return view('dokter.dataabsen_add');
}

public function insert(Request $request)
{
    try{
        $request->validate([
            'tanggal_awal' => 'required',
            'tanggal_akhir' => 'required',
            'alasan' => 'required',
        ],[
            'tanggal_awal' => 'Tanggal awal harus diisi',
            'tanggal_akhir' => 'tanggal akhir harus diisi',
            'alasan' => 'alasan harus diisi',
        ]);

        $user = Auth::user();
        $data = [
            'tanggal_awal' => $request->tanggal_awal,
            'tanggal_akhir' => $request->tanggal_akhir,
            'alasan' => $request->alasan,
            'user_id' => $user->id,
        ];

        $this->izin->addData($data);
        Alert::success('Berhasil!', 'Data Izin berhasil diajukan!');
        return redirect('/dataabsen');
    } catch (Exception $e) {
        dd('simpan data gagal : ', $e);
    }
}

public function destroy($dokter_id)
{
    // // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
    // DB::table('detail_jadwal')->where('dokter_id', $dokter_id)->delete();

    // // Hapus entri terkait di tabel izin
    // DB::table('izin')->where('dokter_id', $dokter_id)->delete();

    // // Cari semua reservasi terkait dengan dokter_id melalui rekam_medik
    // $reservasiIds = DB::table('rekam_medik')->where('dokter_id', $dokter_id)-
    >pluck('reservasi_id');

    // // Hapus entri terkait di tabel rekam_medik
    // DB::table('rekam_medik')->where('dokter_id', $dokter_id)->delete();

    // // Hapus entri terkait di tabel reservasi
    // DB::table('reservasi')->whereIn('reservasi_id', $reservasiIds)->delete();

    // Setelah itu, hapus entri dari tabel dokter
    DB::table('izin dokter')->where('izin_id', $dokter_id)->delete();
}

```

```

        // Menampilkan alert sukses dan mengarahkan kembali ke halaman data dokter
        Alert::success('Berhasil!', 'Data izin berhasil dihapus!');
        return redirect('/dataabsen');
    }

    public function edit($id)
    {
        $izin = DB::table('izin_dokter')->where('izin_id', $id)->first();
        return view('dokter.dataabsen_edit', ['izin' => $izin]);
    }

    public function detail($id)
    {
        $datajadwal = DB::table('detail_jadwal')->where('dokter_id', $id)->get();
        return view('admin.izinjadwal', ['datajadwal' => $datajadwal]);
    }

    public function tolak($id)
    {
        $izin = Izin::where('izin_id', $id)->firstOrFail();

        // Prepare data for reservasi
        $perizinan = [
            'status' => 0,
        ];
        // Update the reservasi
        $izin->update($perizinan);

        // Redirect or return a response
        Alert::success('Berhasil!', 'Data perizinan dokter ditolak !');
        return redirect()->route('owner.izin.index');
    }

    public function terima($id)
    {
        $izin = Izin::where('izin_id', $id)->firstOrFail();

        // Prepare data for reservasi
        $perizinan = [
            'status' => 1,
        ];
        // Update the reservasi
        $izin->update($perizinan);

        // Redirect or return a response
        Alert::success('Berhasil!', 'Data perizinan dokter diterima !');
        return redirect()->route('owner.izin.index');
    }

    public function update(Request $request, $id)
    {
        $request->validate([
            'tanggal_awal' => 'required',
            'tanggal_akhir' => 'required',
            'alasan' => 'required',
        ]);

        $data = [
            'tanggal_awal' => $request->tanggal_awal,
            'tanggal_akhir' => $request->tanggal_akhir,
            'alasan' => $request->alasan,
        ];

        DB::table('izin_dokter')->where('izin_id', $id)->update($data);
        Alert::success('Berhasil!', 'Data pengajuan izin berhasil diupdate!');
        return redirect('/dataabsen');
    }
}

```

## Kode Controller Owner

### 1. PemasukanController.php

```

<?php

namespace App\Http\Controllers\Owner;

use Session;

```

```

use Illuminate\Http\Request;
use App\Models\Reservasi;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use App\Models\Perawatan;
use App\Models\PerawatanReservasi;
use Illuminate\Support\Facades\Auth;

class PemasukanController extends Controller
{
    protected $reservasi;

    public function __construct(reservasi $pemasukan)
    {
        $this->reservasi = $pemasukan;
    }

    public function index()
    {
        $pemasukan = Reservasi::with(['pasien', 'dokter', 'perawatan_reservasi.perawatan'])
        ->where('draft', 0)
        ->get();

        $perawatan_reservasi = PerawatanReservasi::with(['reservasi'])
        ->get();

        $perawatan = Perawatan::get();

        return view('owner.datapemasukan', ["pemasukan"=> $pemasukan, "perawatan_reservasi"=>
        $perawatan_reservasi, "perawatan"=> $perawatan]);
    }
}

```

## 2. PengeluaranController.php

```

<?php

namespace App\Http\Controllers\Owner;

use Session;
use Illuminate\Http\Request;
use App\Models\Pengeluaran;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class PengeluaranController extends Controller
{
    protected $pengeluaran;

    public function __construct(pengeluaran $pengeluaran)
    {
        $this->pengeluaran = $pengeluaran;
    }

    public function index()
    {
        $pengeluaran = [
            'pengeluaran' => $this->pengeluaran->allData(),
        ];
        return view('owner.datapengeluaran', $pengeluaran);
    }

    public function add()
    {
        return view('owner.datapengeluaran_add');
    }

    public function insert(Request $request)
    {
        $request->validate([
            'nama_pengeluaran' => 'required',
            'deskripsi_pengeluaran' => 'required',
            'kategori_pengeluaran' => 'required',
            'jumlah_pengeluaran' => 'required',
            'tanggal' => 'required',
        ],[

```

```

        'nama_pengeluaran.required' => 'nama pengeluaran harus diisi!',
        'deskripsi_pengeluaran.required' => 'estimasi waktu pengeluaran harus diisi!',
        'kategori_pengeluaran' => 'jenis pengeluaran harus diisi!',
        'jumlah_pengeluaran' => 'jumlah pengeluaran harus diisi!',
        'tanggal' => 'tanggal harus diisi!',

    ]);

    $user = Auth::user();

    $data = [
        'nama_pengeluaran' => $request->nama_pengeluaran,
        'deskripsi_pengeluaran' => $request->deskripsi_pengeluaran,
        'kategori_pengeluaran' => $request->kategori_pengeluaran,
        'jumlah_pengeluaran' => $request->jumlah_pengeluaran,
        'tanggal' => $request->tanggal,
        'user_id' => $user->id,
    ];

    $this->pengeluaran->addData($data);
    Alert::success('Berhasil!', 'Data pengeluaran berhasil ditambahkan!');
    return redirect('/datapengeluaran');
}

public function destroy($pengeluaran_id)
{
    // Setelah itu, hapus entri dari tabel pengeluaran
    DB::table('pengeluaran')->where('pengeluaran_id', $pengeluaran_id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data pengeluaran
    Alert::success('Berhasil!', 'Data pengeluaran berhasil dihapus!');
    return redirect('/datapengeluaran');
}

public function edit($id)
{
    $pengeluaran = DB::table('pengeluaran')->where('pengeluaran_id', $id)->first();
    return view('owner.datapengeluaran_edit', ['pengeluaran' => $pengeluaran]);
}

public function update(Request $request, $id)
{
    $request->validate([
        'nama_pengeluaran' => 'required',
        'deskripsi_pengeluaran' => 'required',
        'kategori_pengeluaran' => 'required',
        'jumlah_pengeluaran' => 'required',
        'tanggal' => 'required',
    ]);

    $data = [
        'nama_pengeluaran' => $request->nama_pengeluaran,
        'deskripsi_pengeluaran' => $request->deskripsi_pengeluaran,
        'kategori_pengeluaran' => $request->kategori_pengeluaran,
        'jumlah_pengeluaran' => $request->jumlah_pengeluaran,
        'tanggal' => $request->tanggal,
    ];

    DB::table('pengeluaran')->where('pengeluaran_id', $id)->update($data);
    Alert::success('Berhasil!', 'Data pengeluaran berhasil diupdate!');
    return redirect('/datapengeluaran');
}
}

```

### 3. PerawatanController.php

```

<?php

namespace App\Http\Controllers\Owner;

use Session;
use Illuminate\Http\Request;
use App\Models\Perawatan;
use Illuminate\Support\Facades\DB;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use Illuminate\Support\Facades\Auth;

class PerawatanController extends Controller
{
    protected $perawatan;
}

```

```

public function __construct(Perawatan $perawatan)
{
    $this->perawatan = $perawatan;
}

public function index()
{
    $perawatan = [
        'perawatan' => $this->perawatan->allData(),
    ];
    return view('owner.datalayanan', $perawatan);
}

public function add()
{
    return view('owner.datalayanan_add');
}

public function insert(Request $request)
{
    $request->validate([
        'jenis_Perawatan' => 'required',
        'estimasi_waktu_perawatan' => 'required',
        'harga' => 'required',
    ],[
        'jenis_Perawatan.required' => 'jenis perawatan harus diisi!',
        'estimasi_waktu_perawatan.required' => 'estimasi waktu perawatan harus diisi!',
        'harga.required' => 'harga harus diisi!',
    ]);

    $data = [
        'jenis_Perawatan' => $request->jenis_Perawatan,
        'estimasi_waktu_perawatan' => $request->estimasi_waktu_perawatan,
        'harga' => $request->harga,
    ];

    $this->perawatan->addData($data);
    Alert::success('Berhasil!', 'Data perawatan berhasil ditambahkan!');
    return redirect('/datalayanan');
}

public function destroy($layanan_id)
{
    // Setelah itu, hapus entri dari tabel perawatan
    DB::table('perawatan')->where('perawatan_id', $layanan_id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data perawatan
    Alert::success('Berhasil!', 'Data perawatan berhasil dihapus!');
    return redirect('/datalayanan');
}

public function edit($id)
{
    $perawatan = DB::table('perawatan')->where('perawatan_id', $id)->first();
    return view('owner.datalayanan_edit', ['perawatan' => $perawatan]);
}

public function update(Request $request, $id)
{
    $request->validate([
        'jenis_Perawatan' => 'required',
        'estimasi_waktu_perawatan' => 'required',
        'harga' => 'required',
    ]);

    $data = [
        'jenis_Perawatan' => $request->jenis_Perawatan,
        'estimasi_waktu_perawatan' => $request->estimasi_waktu_perawatan,
        'harga' => $request->harga,
    ];

    DB::table('perawatan')->where('perawatan_id', $id)->update($data);
    Alert::success('Berhasil!', 'Data perawatan berhasil diupdate!');
    return redirect('/datalayanan');
}
}

```

### Kode Controller Pasien

## 1. ReservasiPasienController.php

```
<?php

namespace App\Http\Controllers\Pasien;

use Session;
use App\Models\Pasien;
use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Lokasi;
use App\Models\Perawatan;
use App\Models\Reservasi;
use App\Models\RekamMedik;
use App\Models\Dokter;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use App\Models\DetailJadwal;
use Illuminate\Support\Carbon;
use Exception;
use Illuminate\Support\Facades\Log;
use Illuminate\Validation\ValidationException;
use DateTime;
use DateInterval;

class ReservasiPasienController extends Controller
{
    protected $user;

    public function __construct(User $user)
    {
        $this->user = $user;
    }

    public function index()
    {
        $userid = Auth::user();
        $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('pasien.user_id', $userid->id)
        ->where('status_penginput', 0)
        ->select('reservasi_rekam_medik.*',
            'lokasi.*',
            "pasien.*",
            'dokter.nama as nama_dokter')->get();

        $reservasi_admin = DB::table("reservasi_rekam_medik")
        ->join("pasien", "reservasi_rekam_medik.pasien_id", "=", "pasien.pasien_id")->get();

        return view('pasien.reservasipasien', compact('reservasi_rekam_medik'));
    }

    public function pasien()
    {
        $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('draft', 1)
        ->select('reservasi_rekam_medik.*',
            'lokasi.*',
            "pasien.*",
            'dokter.nama as nama_dokter')->get();

        return view('admin.reservasipasien', compact('reservasi_rekam_medik'));
    }

    public function add()
    {
        $lokasi = Lokasi::all();
        $perawatan = Perawatan::all();
    }
}
```

```

$dokter = Dokter::all();

$tomorrow = Carbon::tomorrow()->format('Y-m-d');

// Generate available timeslots
$timeslots = [];
for ($hour = 10; $hour < 21; $hour++) {
    for ($minute = 0; $minute < 60; $minute += 30) {
        $start = Carbon::createFromTime($hour, $minute);
        $end = $start->copy()->addMinutes(30);
        $timeslots[] = [
            'start' => $start->format('H:i'),
            'end' => $end->format('H:i')
        ];
    }
}

return view('pasien.reservasipasien_add', compact('lokasi', 'dokter', 'perawatan',
'tomorrow', 'timeslots'));
}

public function insert(Request $request)
{
    try {
        $validatedData = $request->validate([
            'nama' => 'required|string|max:255',
            'tempat_lahir' => 'required|string|max:255',
            'tanggal_lahir' => 'required|date',
            'pekerjaan' => 'required|string|max:255',
            'alamat' => 'required|string',
            'no_Telp' => 'required|string|max:15',
            'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
            'dokter_id' => 'required|integer',
            'tanggal' => 'required|date|after:today',
            'jam_mulai' => 'required',
            'perawatan_id' => 'required|array|max:2',
            'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
        ]);

        $tanggal = Carbon::parse($validatedData['tanggal'])->format('Y-m-d');

        // Calculate total estimated time
        $totalEstimasi = 0;
        foreach ($validatedData['perawatan_id'] as $perawatanId) {
            $perawatan = Perawatan::find($perawatanId);
            $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
        }

        $startTime = new DateTime($validatedData['jam_mulai']);
        $endTime = (clone $startTime)->modify("+{$totalEstimasi} minutes")->format('H:i');

        // Check for time conflicts
        $conflictingReservation = Reservasi::where('tanggal', $tanggal)
            ->where(function ($query) use ($validatedData, $endTime) {
                $query->whereBetween('jam_mulai', [$validatedData['jam_mulai'], $endTime])
                    ->orWhereBetween('jam_selesai', [$validatedData['jam_mulai'],
$endTime]);
            })
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai',
[$validatedData['jam_mulai']])
            ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai', [$endTime]);
        if ($conflictingReservation->exists()) {
            return redirect()->back();
        }

        // Create new reservation
        $user_id = Auth::id();

        $pasien = Pasien::create([
            'nama' => $validatedData['nama'],
            'tempat_lahir' => $validatedData['tempat_lahir'],
            'tanggal_lahir' => $validatedData['tanggal_lahir'],
            'pekerjaan' => $validatedData['pekerjaan'],
            'alamat' => $validatedData['alamat'],
            'no_Telp' => $validatedData['no_Telp'],
            'user_id' => $user_id,
        ]);

        $reservasi_rekam_medik = Reservasi::create([
            'pasien_id' => $pasien['pasien_id'],
            'lokasi_id' => $validatedData['lokasi_id'],

```

```

        'dokter_id' => $validatedData['dokter_id'],
        'tanggal' => $tanggal,
        'jam_mulai' => $validatedData['jam_mulai'],
        'jam_selesai' => $endTime,
        'status_penginput' => 0
    ]]);

    $syncData = [];
    foreach ($validatedData['perawatan_id'] as $perawatanId) {
        $perawatan = Perawatan::find($perawatanId);
        $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
        $syncData[$perawatanId] = [
            'harga' => $perawatan->harga,
            'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
        ];
    }

    $reservasi_rekam_medik->perawatan()->attach($syncData);

    Alert::success('Success', 'Reservasi berhasil ditambahkan!');
    return redirect()->route('pasien.history');
} catch (Exception $e) {
    dd('simpan data gagal : ', $e);
}
}

public function getBookedTimes(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $tanggal = $request->query('tanggal');
    $tanggalDate = Carbon::parse($tanggal);

    $bookedTimes = Reservasi::where('dokter_id', $dokter_id)
        ->where('tanggal', $tanggalDate)
        ->get(['jam_mulai', 'jam_selesai']);

    return response()->json($bookedTimes);
}

public function getDokterByLokasi(Request $request)
{
    $lokasi_id = $request->query('lokasi_id');
    $dokter = Dokter::where('lokasi_id', $lokasi_id)->get();
    return response()->json($dokter);
}

public function getAvailableDays(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)->pluck('hari');
    return response()->json($jadwals);
}

public function getSesiByDokterAndHari(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $hari = $request->query('hari');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)
        ->where('hari', $hari)
        ->pluck('sesi');
    return response()->json($jadwals);
}

public function destroy($id)
{
    // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
    DB::table('reservasi_rekam_medik')->where('reservasi_id', $id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data user
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik berhasil dihapus!');
    return redirect('/reservasi/admin');
}

public function terima($id)
{
    $reservasi_rekam_medik = Reservasi::where('reservasi_id', $id)->firstOrFail();

    // Prepare data for reservasi_rekam_medik

```



```

$reservasiData = [
    'draft' => 0,
];
// Update the reservasi_rekam_medik
$reservasi_rekam_medik->update($reservasiData);

// Redirect or return a response
Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien diterima !');
return redirect()->route('reservasi.index');
}

public function edit($id)
{
    $lokasi = Lokasi::all();
    $perawatan = Perawatan::all();
    $dokter = Dokter::all();
    $reservasi = Reservasi::with(['pasien', 'dokter', 'lokasi'])
        ->where('reservasi_id', $id)
        ->first();

    $tomorrow = Carbon::tomorrow()->format('Y-m-d');

    // Generate available timeslots
    $timeslots = [];
    for ($hour = 10; $hour < 21; $hour++) {
        for ($minute = 0; $minute < 60; $minute += 30) {
            $start = Carbon::createFromTime($hour, $minute);
            $end = $start->copy()->addMinutes(30);
            $timeslots[] = [
                'start' => $start->format('H:i'),
                'end' => $end->format('H:i')
            ];
        }
    }

    return view('admin.reservasiadmin_edit', compact('lokasi', 'dokter',
'perawatan', 'reservasi', 'tomorrow'));
}

public function update(Request $request, $id)
{
    try {
        $validatedData = $request->validate([
            'tanggal' => 'required|date|after:today',
            'jam_mulai' => 'required',
            'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
            'dokter_id' => 'required|integer',
            'perawatan_id' => 'required|array|max:2',
            'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
        ]);

        // Find existing reservation
        $reservasi_rekam_medik = Reservasi::findOrFail($id);

        // Check for time conflicts
        $conflictingReservation = Reservasi::where('tanggal', $validatedData['tanggal'])
            ->where('jam', $validatedData['jam_mulai'])
            ->where('reservasi_id', '!=', $id)
            ->first();

        if ($conflictingReservation) {
            return redirect()->back()->withErrors(['jam_mulai' => 'The selected time is
already booked.'])->withInput();
        }

        // Update reservation data
        $reservasi_rekam_medik->update([
            'lokasi_id' => $validatedData['lokasi_id'],
            'dokter_id' => $validatedData['dokter_id'],
            'status_penginput' => "1",
            'tanggal' => $validatedData['tanggal'],
            'jam' => $validatedData['jam_mulai'],
        ]);

        $reservasi_rekam_medik->perawatan()->sync($validatedData['perawatan_id']);
    }
}

```

```

        $perawatanData = Perawatan::whereIn('perawatan_id',
$validatedData['perawatan_id'])->get();
        $syncData = [];
        foreach ($perawatanData as $perawatan) {
            $syncData[$perawatan->id] = [
                'harga' => $perawatan->harga,
                'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
            ];
        }

        $reservasi_rekam_medik->perawatan()->update($syncData);

        Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien berhasil
diperbarui!');
        return redirect()->route('reservasi.index');
    } catch (Exception $e) {
        dd('simpan data gagal : ', $e);
    }
}
}

```

## Kode Controller Lainnya

### 1. ProfileController.php

```

<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProfileUpdateRequest;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Redirect;
use Illuminate\View\View;

class ProfileController extends Controller
{
    /**
     * Display the user's profile form.
     */
    public function edit(Request $request): View
    {
        return view('profile.edit', [
            'user' => $request->user(),
        ]);
    }

    /**
     * Update the user's profile information.
     */
    public function update(ProfileUpdateRequest $request): RedirectResponse
    {
        $request->user()->fill($request->validated());

        if ($request->user()->isDirty('email')) {
            $request->user()->email_verified_at = null;
        }

        $request->user()->save();

        return Redirect::route('profile.edit')->with('status', 'profile-updated');
    }

    /**
     * Delete the user's account.
     */
    public function destroy(Request $request): RedirectResponse
    {
        $request->validateWithBag('userDeletion', [
            'password' => ['required', 'current_password'],

```

```

    });

    $user = $request->user();

    Auth::logout();

    $user->delete();

    $request->session()->invalidate();
    $request->session()->regenerateToken();

    return Redirect::to('/');
}
}

```

## 2. WhatsAppController.php

```

<?php

namespace App\Http\Controllers;

use Twilio\Rest\Client;
use Illuminate\Http\Request;
use RealRashid\SweetAlert\Facades\Alert;

class WhatsAppController extends Controller
{
    public function sendWhatsAppMessage(Request $request)
    {
        $twilioSid = "AC11c4912c70a2167cd628621406eefef3";
        $twilioToken = "ffef5f0a4fee7c1bcb7cd8696d2265d";
        $twilioWhatsAppNumber = "whatsapp:+14155238886";
        $recipientNumber = $request->input('phone'); // Ambil nomor dari request
        $message = $request->input('message'); // Ambil pesan dari request

        $twilio = new Client($twilioSid, $twilioToken);

        try {
            $twilio->messages->create(
                "whatsapp:$recipientNumber",
                [
                    "from" => $twilioWhatsAppNumber,
                    "body" => $message,
                ]
            );

            Alert::success('Success', 'Pesan berhasil dikirim');
            return redirect()->route('reservasi.index');
        } catch (\Exception $e) {
            return response()->json(['error' => $e->getMessage()], 500);
        }
    }
}

```

### Kode Controller Pasien

## 3. ReservasiPasienController.php

```

<?php

namespace App\Http\Controllers\Pasien;

use Session;
use App\Models\Pasien;
use Illuminate\Http\Request;
use App\Models\User;
use App\Models\Lokasi;
use App\Models\Perawatan;
use App\Models\Reservasi;
use App\Models\RekamMedik;
use App\Models\Dokter;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;
use RealRashid\SweetAlert\Facades\Alert;
use App\Http\Controllers\Controller;
use App\Models\DetailJadwal;
use Illuminate\Support\Carbon;
use Exception;
use Illuminate\Support\Facades\Log;

```

```

use Illuminate\Validation\ValidationException;
use DateTime;
use DateInterval;

class ReservasiPasienController extends Controller
{
    protected $user;

    public function __construct(User $user)
    {
        $this->user = $user;
    }

    public function index()
    {
        $userid = Auth::user();
        $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('pasien.user_id', $userid->id)
        ->where('status_penginput', 0)
        ->select('reservasi_rekam_medik.*',
            'lokasi.*',
            "pasien.*",
            'dokter.nama as nama_dokter')->get();

        $reservasi_admin = DB::table("reservasi_rekam_medik")
        ->join("pasien", "reservasi_rekam_medik.pasien_id", "=", "pasien.pasien_id")->get();

        return view('pasien.reservasipasien', compact('reservasi_rekam_medik'));
    }

    public function pasien()
    {
        $reservasi_rekam_medik = DB::table('reservasi_rekam_medik')
        ->join('pasien', 'reservasi_rekam_medik.pasien_id', '=', 'pasien.pasien_id')
        ->join('lokasi', 'reservasi_rekam_medik.lokasi_id', '=', 'lokasi.lokasi_id')
        ->join("dokter", "reservasi_rekam_medik.dokter_id", "=", "dokter.dokter_id")
        ->where('draft', 1)
        ->select('reservasi_rekam_medik.*',
            'lokasi.*',
            "pasien.*",
            'dokter.nama as nama_dokter')->get();

        return view('admin.reservasipasien', compact('reservasi_rekam_medik'));
    }

    public function add()
    {
        $lokasi = Lokasi::all();
        $perawatan = Perawatan::all();
        $dokter = Dokter::all();

        $tomorrow = Carbon::tomorrow()->format('Y-m-d');

        // Generate available timeslots
        $timeslots = [];
        for ($hour = 10; $hour < 21; $hour++) {
            for ($minute = 0; $minute < 60; $minute += 30) {
                $start = Carbon::createFromTime($hour, $minute);
                $end = $start->copy()->addMinutes(30);
                $timeslots[] = [
                    'start' => $start->format('H:i'),
                    'end' => $end->format('H:i')
                ];
            }
        }

        return view('pasien.reservasipasien_add', compact('lokasi', 'dokter', 'perawatan',
            'tomorrow', 'timeslots'));
    }

    public function insert(Request $request)
    {
        try {

```

```

$validatedData = $request->validate([
    'nama' => 'required|string|max:255',
    'tempat_lahir' => 'required|string|max:255',
    'tanggal_lahir' => 'required|date',
    'pekerjaan' => 'required|string|max:255',
    'alamat' => 'required|string',
    'no_Telp' => 'required|string|max:15',
    'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
    'dokter_id' => 'required|integer',
    'tanggal' => 'required|date|after:today',
    'jam_mulai' => 'required',
    'perawatan_id' => 'required|array|max:2',
    'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
]);

$tanggal = Carbon::parse($validatedData['tanggal'])->format('Y-m-d');

// Calculate total estimated time
$totalEstimasi = 0;
foreach ($validatedData['perawatan_id'] as $perawatanId) {
    $perawatan = Perawatan::find($perawatanId);
    $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
}

$startTime = new DateTime($validatedData['jam_mulai']);
$endTime = (clone $startTime)->modify("+{$totalEstimasi} minutes")->format('H:i');

// Check for time conflicts
$conflictingReservation = Reservasi::where('tanggal', $tanggal)
->where(function ($query) use ($validatedData, $endTime) {
    $query->whereBetween('jam_mulai', [$validatedData['jam_mulai'], $endTime])
        ->orWhereBetween('jam_selesai', [$validatedData['jam_mulai'],
$endTime]);
    ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai',
[$validatedData['jam_mulai']])
        ->orWhereRaw('? BETWEEN jam_mulai AND jam_selesai', [$endTime]);
    })
->exists();

// Create new reservation
$user_id = Auth::id();

$pasien = Pasien::create([
    'nama' => $validatedData['nama'],
    'tempat_lahir' => $validatedData['tempat_lahir'],
    'tanggal_lahir' => $validatedData['tanggal_lahir'],
    'pekerjaan' => $validatedData['pekerjaan'],
    'alamat' => $validatedData['alamat'],
    'no_Telp' => $validatedData['no_Telp'],
    'user_id' => $user_id,
]);

$reservasi_rekam_medik = Reservasi::create([
    'pasien_id' => $pasien['pasien_id'],
    'lokasi_id' => $validatedData['lokasi_id'],
    'dokter_id' => $validatedData['dokter_id'],
    'tanggal' => $tanggal,
    'jam_mulai' => $validatedData['jam_mulai'],
    'jam_selesai' => $endTime,
    'status_penginput' => 0
]);

$syncData = [];
foreach ($validatedData['perawatan_id'] as $perawatanId) {
    $perawatan = Perawatan::find($perawatanId);
    $totalEstimasi += $perawatan->estimasi_waktu_perawatan;
    $syncData[$perawatanId] = [
        'harga' => $perawatan->harga,
        'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
    ];
}

$reservasi_rekam_medik->perawatan()->attach($syncData);

Alert::success('Success', 'Reservasi berhasil ditambahkan!');
return redirect()->route('pasien.history');
} catch (Exception $e) {
    dd('simpan data gagal : ', $e);
}

```

```

}

public function getBookedTimes(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $tanggal = $request->query('tanggal');
    $tanggalDate = Carbon::parse($tanggal);

    $bookedTimes = Reservasi::where('dokter_id', $dokter_id)
        ->where('tanggal', $tanggalDate)
        ->get(['jam_mulai', 'jam_selesai']);

    return response()->json($bookedTimes);
}

public function getDokterByLokasi(Request $request)
{
    $lokasi_id = $request->query('lokasi_id');
    $dokter = Dokter::where('lokasi_id', $lokasi_id)->get();
    return response()->json($dokter);
}

public function getAvailableDays(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)->pluck('hari');
    return response()->json($jadwals);
}

public function getSesiByDokterAndHari(Request $request)
{
    $dokter_id = $request->query('dokter_id');
    $hari = $request->query('hari');
    $jadwals = DetailJadwal::where('dokter_id', $dokter_id)
        ->where('hari', $hari)
        ->pluck('sesi');
    return response()->json($jadwals);
}

public function destroy($id)
{
    // Hapus entri terkait di tabel detail_jadwal terlebih dahulu
    DB::table('reservasi_rekam_medik')->where('reservasi_id', $id)->delete();

    // Menampilkan alert sukses dan mengarahkan kembali ke halaman data user
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik berhasil dihapus!');
    return redirect('/reservasi/admin');
}

public function terima($id)
{
    $reservasi_rekam_medik = Reservasi::where('reservasi_id', $id)->firstOrFail();

    // Prepare data for reservasi_rekam_medik
    $reservasiData = [
        'draft' => 0,
    ];
    // Update the reservasi_rekam_medik
    $reservasi_rekam_medik->update($reservasiData);

    // Redirect or return a response
    Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien diterima !');
    return redirect()->route('reservasi.index');
}

public function edit($id)
{
    $lokasi = Lokasi::all();
    $perawatan = Perawatan::all();
    $dokter = Dokter::all();
    $reservasi = Reservasi::with(['pasien', 'dokter', 'lokasi'])
        ->where('reservasi_id', $id)
        ->first();

    $tomorrow = Carbon::tomorrow()->format('Y-m-d');

```

```

// Generate available timeslots
$timeslots = [];
for ($hour = 10; $hour < 21; $hour++) {
    for ($minute = 0; $minute < 60; $minute += 30) {
        $start = Carbon::createFromTime($hour, $minute);
        $end = $start->copy()->addMinutes(30);
        $timeslots[] = [
            'start' => $start->format('H:i'),
            'end' => $end->format('H:i')
        ];
    }
}

return view('admin.reservasiadmin_edit', compact('lokasi', 'dokter',
'perawatan', 'reservasi', 'tomorrow'));
}

public function update(Request $request, $id)
{
    try {
        $validatedData = $request->validate([
            'tanggal' => 'required|date|after:today',
            'jam_mulai' => 'required',
            'lokasi_id' => 'required|integer|exists:lokasi,lokasi_id',
            'dokter_id' => 'required|integer',
            'perawatan_id' => 'required|array|max:2',
            'perawatan_id.*' => 'integer|exists:perawatan,perawatan_id',
        ]);

        // Find existing reservation
        $reservasi_rekam_medik = Reservasi::findOrFail($id);

        // Check for time conflicts
        $conflictingReservation = Reservasi::where('tanggal', $validatedData['tanggal'])
            ->where('jam', $validatedData['jam_mulai'])
            ->where('reservasi_id', '!=', $id)
            ->first();

        if ($conflictingReservation) {
            return redirect()->back()->withErrors(['jam_mulai' => 'The selected time is
already booked.'])->withInput();
        }

        // Update reservation data
        $reservasi_rekam_medik->update([
            'lokasi_id' => $validatedData['lokasi_id'],
            'dokter_id' => $validatedData['dokter_id'],
            'status_penginput' => "1",
            'tanggal' => $validatedData['tanggal'],
            'jam' => $validatedData['jam_mulai'],
        ]);

        $reservasi_rekam_medik->perawatan()->sync($validatedData['perawatan_id']);
        $perawatanData = Perawatan::whereIn('perawatan_id',
$validatedData['perawatan_id'])->get();
        $syncData = [];
        foreach ($perawatanData as $perawatan) {
            $syncData[$perawatan->id] = [
                'harga' => $perawatan->harga,
                'estimasi_waktu_perawatan' => $perawatan->estimasi_waktu_perawatan,
            ];
        }

        $reservasi_rekam_medik->perawatan()->update($syncData);

        Alert::success('Berhasil!', 'Data reservasi_rekam_medik pasien berhasil
diperbarui!');
        return redirect()->route('reservasi.index');
    } catch (Exception $e) {
        dd('simpan data gagal : ', $e);
    }
}
}

```

## Kode Models

### 1. Admin.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Admin extends Model
{
    use HasFactory;

    protected $table = 'admin';
    protected $primaryKey = 'admin_id';
    public $timestamps = false;

    protected $fillable = [
        'nama',
        'alamat',
        'email',
        'password'
    ];

    public function reservasi()
    {
        return $this->hasMany(Reservasi::class, 'admin_id');
    }

    public function pengeluaran()
    {
        return $this->hasMany(Pengeluaran::class, 'admin_id');
    }
}

```

## 2. DetailDokter.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class DataDokter extends Model
{
    protected $table = 'dokter';

    public function users()
    {
        return $this->hasOne(User::class, 'id', 'id');
    }

    public function allData()
    {
        return DB::table('dokter')->get();
    }

    public function addData($data)
    {
        DB::table('dokter')->insert($data);
    }
}

```

## 3. DetailJadwal.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class DetailJadwal extends Model
{
    use HasFactory;

    protected $table = 'detail_jadwal';
    protected $primaryKey = 'jadwal_id';
    public $timestamps = false;
}

```



```

        protected $fillable = [
            'dokter_id',
            'hari',
            'sesi'
        ];

        public function dokter()
        {
            return $this->belongsTo(Dokter::class, 'dokter_id');
        }

        public function allData()
        {
            return DB::table('detail_jadwal')->get();
        }

        public function addData($data)
        {
            DB::table('detail_jadwal')->insert($data);
        }
    }

```

#### 4. Dokter.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Dokter extends Model
{
    use HasFactory;

    protected $table = 'dokter';
    protected $primaryKey = 'dokter_id';
    public $timestamps = false;

    protected $fillable = [
        'nama',
        'alamat',
        'nomor_hp',
        'lokasi_id',
        'user_id',
    ];

    public function detailJadwal()
    {
        return $this->hasMany(DetailJadwal::class, 'dokter_id');
    }

    public function izin()
    {
        return $this->hasMany(Izin::class, 'dokter_id');
    }

    public function rekamMedik()
    {
        return $this->hasMany(RekamMedik::class, 'dokter_id');
    }
}

```

#### 5. Izin.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Izin extends Model
{
    use HasFactory;

    protected $table = 'izin_dokter';
    protected $primaryKey = 'izin_id';
    public $timestamps = false;

    protected $fillable = [

```

```

        'user_id',
        'tanggal_awal',
        'tanggal_akhir',
        'alasan',
        'status',
    ];

    public function allData()
    {
        return DB::table('izin_dokter')->get();
    }

    public function addData($data)
    {
        DB::table('izin_dokter')->insert($data);
    }

    public function users()
    {
        return $this->belongsTo(User::class, 'user_id');
    }
}

```

## 6. Lokasi.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Lokasi extends Model
{
    use HasFactory;

    protected $table = 'lokasi';
    protected $primaryKey = 'lokasi_id';
    public $timestamps = false;

    protected $fillable = [
        'nama_lokasi',
        'alamat'
    ];

    public function reservasi()
    {
        return $this->hasMany(Reservasi::class, 'lokasi_id');
    }
}

```

## 7. Pasien.php

```

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Pasien extends Model
{
    protected $table = 'pasien';
    protected $primaryKey = 'pasien_id';
    public $timestamps = false;

    protected $fillable = [
        'nama',
        'tempat_lahir',
        'tanggal_lahir',
        'pekerjaan',
        'alamat',
        'no_telp',
        'user_id'
    ];

    public function users()
    {
        return $this->hasOne(User::class, 'id', 'id');
    }
}

```

```

    public function allData()
    {
        return DB::table('pasien')->get();
    }

    public function addData($data)
    {
        DB::table('pasien')->insert($data);
    }

    public function pasien()
    {
        return $this->hasMany(Reservasi::class, 'id', 'id');
    }
}

```

## 8. Pengeluaran.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Pengeluaran extends Model
{
    use HasFactory;

    protected $table = 'pengeluaran';
    protected $primaryKey = 'pengeluaran_id';
    public $timestamps = false;

    protected $fillable = [
        'admin_id',
        'deskripsi_pengeluaran',
        'nama_pengeluaran',
        'kategori_pengeluaran',
        'jumlah_pengeluaran',
        'tanggal'
    ];

    public function admin()
    {
        return $this->belongsTo(Admin::class, 'admin_id');
    }

    public function allData()
    {
        return DB::table('pengeluaran')->get();
    }

    public function addData($data)
    {
        DB::table('pengeluaran')->insert($data);
    }
}

```

## 9. Perawatan.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Perawatan extends Model
{
    use HasFactory;

    protected $table = 'perawatan';
    protected $primaryKey = 'perawatan_id';
    public $timestamps = false;

    protected $fillable = [
        'jenis_Perawatan',
        'harga',
        'estimasi_waktu_perawatan'
    ];
}

```

```

        public function reservasi()
        {
            return $this->belongsToMany(Reservasi::class, 'perawatan_reservasi', 'perawatan_id',
            'reservasi_id');
        }

        public function allData()
        {
            return DB::table('perawatan')->get();
        }

        public function addData($data)
        {
            DB::table('perawatan')->insert($data);
        }
    }
}

```

## 10. PerawatanReservasi.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class PerawatanReservasi extends Model
{
    use HasFactory;

    protected $table = 'perawatan_reservasi';
    protected $primaryKey = 'id';
    public $timestamps = false;

    protected $fillable = [
        'perawatan_id',
        'reservasi_id',
        'harga',
        'estimasi_waktu_perawatan'
    ];

    public function reservasi()
    {
        return $this->belongsTo(Reservasi::class, 'reservasi_id', 'reservasi_id');
    }

    public function perawatan()
    {
        return $this->belongsTo(Perawatan::class, 'perawatan_id', 'perawatan_id');
    }

    public function allData()
    {
        return DB::table('perawatan_reservasi')->get();
    }
}

```

## 11. RekamMedik.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class RekamMedik extends Model
{
    use HasFactory;

    protected $table = 'rekam_medik';
    protected $primaryKey = 'rekam_id';
    public $timestamps = false;

    protected $fillable = [
        'reservasi_id',

```

```

        'dokter_id',
        'perawatan_id',
        'golongan_darah',
        'tekanan_darah',
        'penyakit_jantung',
        'diabetes',
        'hepatitis',
        'penyakit_lainnya',
        'alergi_makanan',
        'alergi_obat',
        'keluhan',
        'gigi',
        'biaya'
    ];

    public function reservasi()
    {
        return $this->belongsTo(Reservasi::class, 'reservasi_id', 'reservasi_id');
    }

    public function dokter()
    {
        return $this->belongsTo(Dokter::class, 'dokter_id');
    }

    public function allData()
    {
        return DB::table('rekam_medik')->get();
    }

    public function addData($data)
    {
        DB::table('rekam_medik')->insert($data);
    }
}

```

## 12. Reservasi.php

```

<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Support\Facades\DB;

class Reservasi extends Model
{
    use HasFactory;

    protected $table = 'reservasi_rekam_medik';
    protected $primaryKey = 'reservasi_id';
    public $timestamps = false;

    protected $fillable = [
        'pasien_id',
        'lokasi_id',
        'dokter_id',
        'admin_id',
        'rekam_medik_id',
        'tanggal',
        'jam_mulai',
        'jam_selesai',
        'golongan_darah',
        'tekanan_darah',
        'penyakit_jantung',
        'diabetes',
        'hepatitis',
        'penyakit_lainnya',
        'alergi_makanan',
        'alergi_obat',
        'keluhan',
        'gigi',
        'draft',
        'status_penginput',
    ];

    public function allData()
    {
        return DB::table('reservasi_rekam_medik')->get();
    }
}

```

```

public function pasien()
{
    return $this->belongsTo(Pasien::class, 'pasien_id');
}

public function dokter()
{
    return $this->belongsTo(Dokter::class, 'dokter_id');
}

public function lokasi()
{
    return $this->belongsTo(Lokasi::class, 'lokasi_id');
}

public function perawatan()
{
    return $this->belongsToMany(Perawatan::class, 'perawatan_reservasi', 'reservasi_id',
'perawatan_id')
        ->withPivot('harga', 'estimasi_waktu_perawatan');
}

// Metode untuk menghitung total harga perawatan
public function getTotalHargaPerawatanAttribute()
{
    return $this->perawatan->sum('pivot.harga');
}

public function user()
{
    return $this->belongsTo(Admin::class, 'id');
}

public function rekamMedik()
{
    return $this->hasOne(RekamMedik::class, 'reservasi_id', 'reservasi_id');
}

public function perawatan_reservasi()
{
    return $this->hasMany(PerawatanReservasi::class, 'reservasi_id', 'reservasi_id');
}

}

```

### 13. User.php

```

<?php

namespace App\Models;

// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Support\Facades\DB;
use Illuminate\Notifications\Notifiable;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'role',
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [

```

```

        'password',
        'remember_token',
    ];

    /**
     * Get the attributes that should be cast.
     *
     * @return array<string, string>
     */
    protected function casts(): array
    {
        return [
            'email_verified_at' => 'datetime',
            'password' => 'hashed',
        ];
    }

    public function allData()
    {
        return DB::table('users')->get();
    }

    public function addData($data)
    {
        $user = User::create($data);
        return $user->id;
    }
}

```

**LAMPIRAN F**  
**PENGUJIAN SISTEM**  
**(LANJUTAN)**



# 1. Admin

## LEMBAR PENGUJIAN APLIKASI

Nama Aplikasi : Sistem Informasi Manajemen Klinik Gigi

Nama Penguji : *Fadel Nela Febrianti*

Hak Akses/Jabatan : Admin

No	Fungsional	Skenario	Hasil yang diharapkan	Hasil	
				Sesuai	Tidak Sesuai
1	Authentikasi dan Pengelolaan akun	Login	Sistem berhasil login dan menampilkan dashboard	✓	
		Ganti Password	Sistem berhasil mengubah password akun	✓	
		Edit Profil	Sistem berhasil mengubah profil akun	✓	
		Hapus Akun	Sistem berhasil menghapus akun user	✓	
2	Lihat Dashboard	Lihat	Sistem berhasil menampilkan dashboard	✓	
		Filter	Sistem berhasil menampilkan informasi yang berbeda dari dashboard	✓	
3	Konfirmasi reservasi dari Pasien	Lihat	Sistem berhasil menampilkan list reservasi yang diajukan pasien	✓	
		Tolak	Sistem berhasil menolak reservasi dari pasien	✓	
		Terima	Sistem berhasil menerima reservasi dari pasien.	✓	
4	Reservasi pasien	Lihat	Sistem berhasil menampilkan list reservasi yang dibuat admin	✓	
		Tambah	Sistem berhasil menambahkan reservasi	✓	
		Edit	Sistem berhasil mengedit informasi reservasi	✓	
5	Mengelola data pasien	Lihat	Sistem berhasil menampilkan list pasien	✓	
		Edit	Sistem berhasil mengedit informasi pasien	✓	
		Hapus	Sistem berhasil menghapus data pasien	✓	
6	Mengelola data	Lihat	Sistem berhasil	✓	

	rekam medik		menampilkan rekam medik pasien	✓	
		Edit	Sistem berhasil mengedit informasi rekam medik pasien	✓	
7	Mengelola data dokter, jadwal dokter	Lihat	Sistem berhasil menampilkan data dokter dan jadwal praktek dokter	✓	
		Tambah	Sistem berhasil menambahkan data dokter dan jadwal praktek dokter	✓	
		Edit	Sistem berhasil mengedit data dokter dan jadwal praktek dokter	✓	
		Hapus	Sistem berhasil menghapus data dokter dan jadwal praktek dokter	✓	
8	Mengelola data user	Lihat	Sistem informasi berhasil menampilkan list user web	✓	
		Tambah	Sistem informasi berhasil menambahkan user	✓	
		Edit	Sistem informasi berhasil mengedit user	✓	
		Hapus	Sistem informasi berhasil menghapus user	✓	

Padang, September 2024

Penguji

(  )  
Faten Nida F.

## 2. Owner

Nama Aplikasi : Sistem Informasi Manajemen Klinik Gigi

Nama Penguji : Ika. Muhammad Atri Pamelha

Hak Akses/Jabatan : Owner

No	Fungsional	Skenario	Hasil yang diharapkan	Hasil	
				Sesuai	Tidak Sesuai
1	Authentikasi dan Pengelolaan akun	Login	Sistem berhasil login dan menampilkan dashboard	✓	
		Ganti Password	Sistem berhasil mengubah password akun	✓	
		Edit Profil	Sistem berhasil mengubah profil akun	✓	
		Hapus Akun	Sistem berhasil menghapus akun user	✓	
2	Lihat Dashboard	Lihat	Sistem berhasil menampilkan dashboard	✓	
		Filter	Sistem berhasil menampilkan informasi yang berbeda dari dashboard	✓	
3	Melihat data profit klinik	Lihat	Sistem berhasil menampilkan profit klinik	✓	
		Filter	Sistem berhasil mengfilter profit berdasarkan bulan	✓	
4	Melihat jumlah kunjungan pasien	Lihat	Sistem berhasil menampilkan jumlah kunjungan pasien	✓	
		Filter	Sistem berhasil mengfilter jumlah kunjungan berdasarkan bulan	✓	
5	Melihat jumlah penanganan perawatan berdasarkan kategori	Lihat	Sistem berhasil menampilkan jumlah penanganan perawatan berdasarkan kategori	✓	
		Filter	Sistem berhasil mengfilter jumlah penanganan perawatan berdasarkan bulan	✓	
6	Melihat performa dokter berdasarkan jumlah pasien	Lihat	Sistem berhasil menampilkan data performa dokter berdasarkan jumlah pasien yang dirawat	✓	
		Filter	Sistem berhasil mengfilter data jumlah pasien yang dirawat dokter	✓	
7	Melihat data	Lihat	Sistem berhasil menampilkan	✓	

	pasien berdasarkan umur		data pasien berdasarkan umurnya	✓	
		Filter	Sistem berhasil mengfilter profit berdasarkan bulan	✓	
8	Melihat Data dokter	Lihat	Sistem berhasil menampilkan list data dokter	✓	
9	Melihat Data Pasien	Lihat	Sistem berhasil menampilkan list data pasien	✓	
10	Melihat Data Pemasukan	Lihat	Sistem berhasil menampilkan list data pemasukan	✓	
11	Memutuskan perizinan dokter	Terima	Sistem berhasil menerima perizinan dokter	✓	
		Tolak	Sistem berhasil menolak perizinan dokter	✓	
12	Mengelola pencatatan pengeluaran klinik	Lihat	Sistem berhasil menampilkan pengeluaran klinik	✓	
		Tambah	Sistem berhasil menambah pengeluaran klinik	✓	
		Edit	Sistem berhasil mengedit pengeluaran klinik	✓	
		Hapus	Sistem berhasil menghapus pengeluaran klinik	✓	
13	Mengelola harga perawatan	Lihat	Sistem berhasil menampilkan harga perawatan	✓	
		Tambah	Sistem berhasil menambahkan harga perawatan	✓	
		Edit	Sistem berhasil mengedit harga perawatan	✓	
		Hapus	Sistem berhasil menghapus harga perawatan		

Padang, September 2024

Penguji



(Dr. Muhammad Datri Rajadkhoni)

### 3. Dokter

Nama Aplikasi : Sistem Informasi Manajemen Klinik Gigi

Nama Penguji : Drg. Fachan Muhammad Nave

Hak Akses/Jabatan : Dokter

No	Fungsional	Skenario	Hasil yang diharapkan	Hasil	
				Sesuai	Tidak Sesuai
1	Authentikasi dan Pengelolaan akun	Login	Sistem berhasil login dan menampilkan dashboard	✓	
		Ganti Password	Sistem berhasil mengubah password akun	✓	
		Edit Profil	Sistem berhasil mengubah profil akun	✓	
		Hapus Akun	Sistem berhasil menghapus akun user	✓	
2	Lihat Jadwal Praktek	Lihat	Sistem berhasil menampilkan jadwal praktek dokter	✓	
3	Mengelola data pasien	Lihat	Sistem berhasil menampilkan list data pasien	✓	
		Edit	Sistem berhasil mengedit list data pasien	✓	
		Hapus	Sistem berhasil menghapus list data pasien	✓	
4	Mengajukan permintaan izin	Lihat	Sistem berhasil menampilkan list permintaan izin yang diajukan	✓	
		Tambah	Sistem berhasil mengajukan permintaan izin ke owner	✓	
		Edit	Sistem berhasil mengedit permintaan izin ke owner	✓	
		Hapus	Sistem berhasil menghapus permintaan izin ke owner	✓	

Padang, September 2024

Penguji



(Drg. Fachan Muhammad Nave)



#### 4. Pasien

Nama Aplikasi : Sistem Informasi Manajemen Klinik Gigi

Nama Penguji : Laras Adinda Putry

Hak Akses/Jabatan : Pasien

No	Fungsional	Skenario	Hasil yang diharapkan	Hasil	
				Sesuai	Tidak Sesuai
1	Authentikasi dan Pengelolaan akun	Login	Sistem berhasil login dan menampilkan dashboard	✓	
		Ganti Password	Sistem berhasil mengubah password akun	✓	
		Edit Profil	Sistem berhasil mengubah profil akun	✓	
		Hapus Akun	Sistem berhasil menghapus akun user	✓	
2	Registrasi Akun	Registrasi	Sistem berhasil melakukan registrasi akun	✓	
3	Reservasi klinik	Tambah	Sistem berhasil mengajukan reservasi ke klinik	✓	
4	Melihat Riwayat reservasi	Lihat	Sistem berhasil melihat riwayat reservasi yang sudah pernah diajukan	✓	

Padang, September 2024

Penguji



(LARAS ADINDA PUTRY)

**LAMPIRAN G**  
**DATA DOKUMEN**

## 1. Berita Acara Wawancara

### Berita Acara Wawancara

Waktu wawancara : 02 Oktober 2023  
Lokasi Wawancara : Cafe Kopi Boasfery, Padang

#### Profil Pewawancara

Nama : Alif Abdul Rauf  
NIM : 2011522024

#### Profil Narasumber

Nama : Drs. Muhammad Fitri Ramdhani  
Jabatan : Owner

#### Hasil Wawancara

- Pertanyaan** Bagaimana proses pendaftaran untuk bisa melakukan perawatan pada klinik xenon dental house?
- Jawaban** Saat ini pendaftaran untuk perawatan bisa dilakukan dengan 2 cara. Pertama pasien bisa menghubungi klinik melalui no whatsapp klinik gigi xenon dental house. Pasien akan diminta untuk mengisi format pesan yang sudah disediakan dari klinik, jika sudah dikirim dan telah sesuai maka pasien akan terdaftar untuk reservasinya. Kedua pasien bisa langsung mendatangi klinik untuk melakukan reservasi. Untuk reservasi ini akan dibantu dengan admin dari klinik ini secara langsung.
- Pertanyaan** Proses apa yang terjadi setelah dilakukannya reservasi?
- Jawaban** Pada hari perawatan, admin akan mengingatkan pasien 1 jam sebelum waktu perawatan untuk datang ke klinik sesuai dengan waktu yang sudah direservasi. Saat pasien sudah datang, admin akan mengisi data pribadi pasien pada lembar awal rekam medik. Setelah itu pasien diarahkan masuk ke ruangan perawatan untuk dilakukan perawatan oleh dokter. Dokter akan memeriksa



pasien, bertanya kepada pasien terkait kondisi pasien, riwayat penyakit dan dokter akan mengisikan rekam medik pasien. Setelah itu akan dilakukan perawatan pasien, perawatan ini bisa berupa cabut gigi, behel, tambal gigi, *scaling*, *bleaching*, gigi tiruan. Setelah selesai admin akan memeriksa rekam medik pasien dan meminta pasien untuk melakukan pembayaran. Jika dari proses perawatan reservasi yang dilakukan memerlukan perawatan lebih lanjut maka pasien akan direservasikan ulang untuk perawatan selanjutnya.

Pertanyaan Apa saja peran yang ada pada klinik xenon dental house?

Jawaban Di klinik ini terbagi menjadi 2 yaitu ada bagian administrasi dan tenaga medis. Untuk administrasi adalah orang yang bertanggung jawab mengelola aspek administratif klinik, seperti pengelolaan jadwal, sosial media, dan resepsionis. Untuk tenaga medis terdiri dari dokter dan asisten dokter yang akan melakukan perawatan kepada pasien. Pada klinik ini juga terdapat dokter pengganti yang akan menggantikan dokter tetap di klinik ini jika dokter tetap sedang izin untuk tidak bekerja kepada owner.

Pertanyaan Apa saja peran yang ada pada klinik xenon dental house?

Jawaban Di klinik ini terbagi menjadi 2 yaitu ada bagian administrasi dan tenaga medis. Untuk administrasi adalah orang yang bertanggung jawab mengelola aspek administratif klinik, seperti pengelolaan jadwal, sosial media, dan resepsionis. Untuk tenaga medis terdiri dari dokter dan asisten dokter yang akan melakukan perawatan kepada pasien. Pada klinik ini juga terdapat dokter pengganti yang akan menggantikan dokter tetap di klinik ini jika dokter tetap sedang izin untuk tidak bekerja kepada owner.

Pertanyaan Dari proses bisnis yang ada ini apakah ada fitur tambahan yang dirasa dibutuhkan untuk klinik xenon dental house?

Jawaban Dibutuhkan rekapan atau ringkasan data yang berisi data kategori penanganan, pasien dan keuangan. Sehingga data yang bersangkutan bisa dipantau dan bisa dijadikan data untuk membantu dalam pengambilan keputusan.

Padang, Oktober 2023

Narasumber



(Dr. Muhammad Atri) Ramadhani

## 2. Surat Pernyataan Owner

### SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : drg. Muhammad Afri Esmadhen  
Jabatan : Owner

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama :

Nama : Alif Abdul Rauf  
NIM : 2011522024  
Departemen/Fakultas : Sistem Informasi/Teknologi Informasi  
PTN BH : Universitas Andalas

Terkait Tugas Akhir "Rancang Bangun Sistem Informasi Manajemen Klinik Gigi Studi Kasus : Klinik Gigi Xenon Dental House" bahwa aplikasi yang dibangun telah sesuai dengan bisnis proses yang ada dan sesuai kebutuhan klinik gigi xenon dental house departemen Sistem Informasi.

Demikian surat pernyataan ini saya berikan untuk digunakan dengan sebagaimana mestinya.

Padang, September 2024

Penguji



(drg. Muhammad Afri Esmadhen)

### 3. Surat Pernyataan Admin

#### SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : Faten Nela Febrianti  
Jabatan : Admin

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama :

Nama : Alif Abdul Rauf  
NIM : 2011522024  
Departemen/Fakultas : Sistem Informasi/Teknologi Informasi  
PTN BH : Universitas Andalas

Terkait Tugas Akhir "Rancang Bangun Sistem Informasi Manajemen Klinik Gigi Studi Kasus : Klinik Gigi Xenon Dental House" bahwa aplikasi yang dibangun telah sesuai dengan bisnis proses yang ada dan sesuai kebutuhan klinik gigi xenon dental house departemen Sistem Informasi.

Demikian surat pernyataan ini saya berikan untuk digunakan dengan sebagaimana mestinya.

Padang, September 2024

Penguji

  
( Faten Nela Febrianti

#### 4. Surat Pernyataan Dokter

##### SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : Drg. Fathah Muhammad Nouvz  
Jabatan : Dokter

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama :

Nama : Alif Abdul Rauf  
NIM : 2011522024  
Departemen/Fakultas : Sistem Informasi/Teknologi Informasi  
PTN BH : Universitas Andalas

Terkait Tugas Akhir "Rancang Bangun Sistem Informasi Manajemen Klinik Gigi Studi Kasus : Klinik Gigi Xenon Dental House" bahwa aplikasi yang dibangun telah sesuai dengan bisnis proses yang ada dan sesuai kebutuhan klinik gigi xenon dental house departemen Sistem Informasi.

Demikian surat pernyataan ini saya berikan untuk digunakan dengan sebagaimana mestinya.

Padang, September 2024

Penguji



(Drg. Fathah Muhammad Nouvz)

## 5. Surat Pernyataan Pasien

### SURAT PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : Lutas Adinda Putty  
Jabatan : Pasien

Menyatakan bahwa mahasiswa Perguruan Tinggi Negeri Berbadan Hukum yang bernama :

Nama : Alif Abdul Rauf  
NIM : 2011522024  
Departemen/Fakultas : Sistem Informasi/Teknologi Informasi  
PTN BH : Universitas Andalas

Terkait Tugas Akhir "**Rancang Bangun Sistem Informasi Manajemen Klinik Gigi Studi Kasus : Klinik Gigi Xenon Dental House**" bahwa aplikasi yang dibangun telah sesuai dengan bisnis proses yang ada dan sesuai kebutuhan klinik gigi xenon dental house departemen Sistem Informasi.

Demikian surat pernyataan ini saya berikan untuk digunakan dengan sebagaimana mestinya.

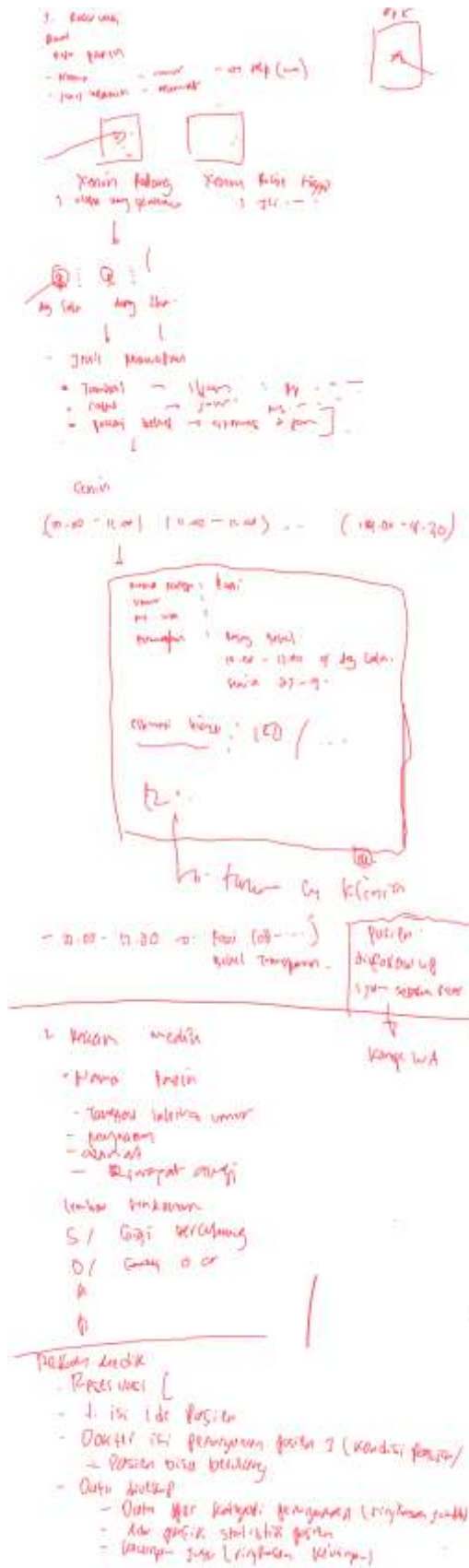
Padang, September 2024

Penguji



(LUTAS ADINDA PUTTY )

## 6. Bukti hasil *brainstorming*



## 7. Buku Pengeluaran

No		Uraian		Jumlah / Satuan	
No		Uraian		Jumlah / Satuan	
1	5	HARDMAN K (2)			
2	5	IMBROSE D-25 (1)			
		HARDMAN D-25 (1)			
		(MATERIAL DAN BAHAN)			
		GLOBAL DENT			
3	5	APPROXIMATE			
		Jumlah dari 2 ke 2			15.000
		Jumlah dari 2 ke 2			15.000
		Jumlah dari 2 ke 2			15.000
		TOTAL = Rp 150.000			
4	5	APPROXIMATE			
		Jumlah dari 2 ke 2			15.000
		Jumlah dari 2 ke 2			15.000
		Jumlah dari 2 ke 2			15.000
		TOTAL = Rp 150.000			
14	5	Elastis Modul (3)			
		- Soft (2)			
		HARDMAN (2)			
		GLOBAL DENT			
		TOTAL = Rp 150.000			
14	5	Selain (2)			
		APPROXIMATE			
		TOTAL = Rp 80.000			
15	5	ULTRASONIC Scaler D3 LED			
		Rp 2.600.000			
		GLOBAL DENT			
		Selang 4x6 mm 3m			
		NIPV T (3 joint)			
		Rp 77.000			
		GLOBAL DENT			



The screenshot shows a Microsoft Excel spreadsheet titled "LAPORAN HARIAN PENDAPATAN XENON DENTAL HOUSE Minggu 26 Mei 2024". The spreadsheet is organized into columns for patient information, dental services, and financial data. The data is as follows:

No	NAMA PASIEN	GIGI	NAMA PERAWAT	BAHAN TERPAK	BIAYA	BAYAR	SISA	DOKTER	BATERANGGA
1	[REDACTED]	gigitan	[REDACTED]		Rp. 300.000	Rp. 300.000		Rp. 300.000	IT
2	[REDACTED]	gigitan	[REDACTED]		Rp. 250.000	Rp. 250.000		Rp. 250.000	IT
Total					Rp. 550.000	Rp. 550.000			

Below the table, there is a section for signatures and names:


Dokter Peng.	Dg. Alif
Dokter Tang.	Dg. Alif
ASPP	Vania
Admin	Melita

The Excel interface shows the 'Home' tab selected, with various ribbon options like Font, Paragraph, and Styles visible. The status bar at the bottom indicates the file is named 'Laporan Harian Pendapatan.xlsx' and is located in 'C:\Users\user\Documents'.



## 9. Rekam Medik

[illegible]



**KENOS DENTAL HOUSE**  
(CLINIC & STORE)

11, Telukwang Street, Teluk Kuning, Kelantan, Kota Teluk Kuning, Kelantan  
Tel: 015-2743 0000 Fax: 015-2743 0000

---

TEMPERAT (KOR)	REKAMEN DOKTER	PESAWATAN	RIASA	DOKTER
11/01/2011	1. Gigi berlubang 2. Gigi berlubang 3. Gigi berlubang 4. Gigi berlubang 5. Gigi berlubang 6. Gigi berlubang 7. Gigi berlubang 8. Gigi berlubang 9. Gigi berlubang 10. Gigi berlubang	1. Gigi berlubang 2. Gigi berlubang 3. Gigi berlubang 4. Gigi berlubang 5. Gigi berlubang 6. Gigi berlubang 7. Gigi berlubang 8. Gigi berlubang 9. Gigi berlubang 10. Gigi berlubang	1. Gigi berlubang 2. Gigi berlubang 3. Gigi berlubang 4. Gigi berlubang 5. Gigi berlubang 6. Gigi berlubang 7. Gigi berlubang 8. Gigi berlubang 9. Gigi berlubang 10. Gigi berlubang	1. Gigi berlubang 2. Gigi berlubang 3. Gigi berlubang 4. Gigi berlubang 5. Gigi berlubang 6. Gigi berlubang 7. Gigi berlubang 8. Gigi berlubang 9. Gigi berlubang 10. Gigi berlubang