BAB III

PELAKSANAAN KERJA

3.1 Kedudukan dan Koordinasi

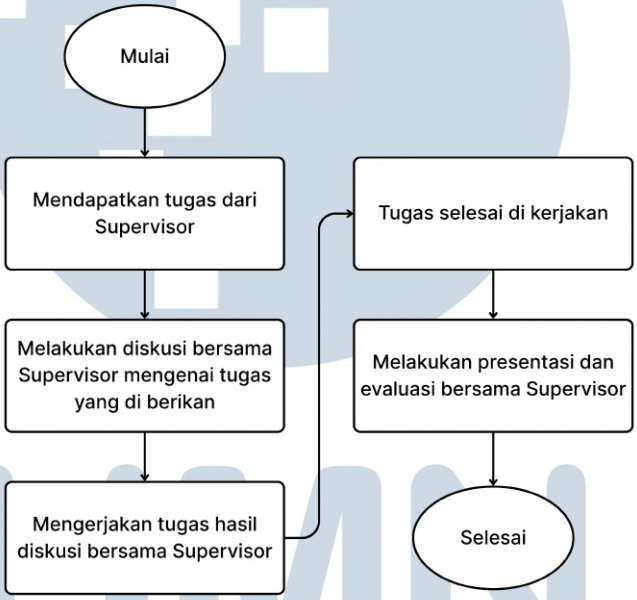
Dalam melakukan pelaksanaan program magang, posisi mahasiswa dalam struktur organisasi divisi *IT* serta mekanisme koordinasi yang diterapkan selama pelaksanaan tugas. Penjelasan mencakup ruang lingkup tanggung jawab posisi *web developer intern*, hubungan fungsional dengan pembimbing lapangan (*supervisor/ SPV IT*) dan pihak terkait lainnya, serta prosedur pengaliran tugas, pelaporan progres, dan mekanisme evaluasi. Tujuan uraian ini adalah memberikan gambaran yang jelas tentang bagaimana pekerjaan diarahkan, dimonitor, dan diselesaikan sehingga kegiatan magang berjalan terstruktur dan memenuhi standar operasional perusahaan.

3.1.1 Kedudukan

Mahasiswa ditempatkan pada divisi IT dengan posisi sebagai *web developer intern* dan berada di bawah pengawasan langsung *Supervisor* divisi IT. Kedudukan ini menempatkan mahasiswa sebagai penanggung jawab utama dalam pengembangan *website* Rumah Sakit Mentari karena tidak terdapat personel lain yang secara khusus menangani sistem web di institusi tersebut. Dengan kondisi tersebut, seluruh pekerjaan teknis terkait pengembangan *website*, mulai dari perancangan antarmuka (*UI/UX)*, implementasi fitur *front-end*, integrasi dengan layanan *back-end*, pengelolaan *database*, hingga pemeliharaan dan perbaikan *bug* dijalankan secara mandiri oleh mahasiswa. Selain itu, mahasiswa juga memiliki tanggung jawab dalam penyusunan dokumentasi teknis, pengelolaan versi kode, serta pelaksanaan pengujian fungsional sebelum sistem dipublikasikan. Kedudukan ini menuntut kemandirian tinggi, kemampuan *problem solving*, serta inisiatif dalam menyelesaikan kendala teknis tanpa dukungan tim khusus. Walaupun dikerjakan secara individual, seluruh hasil kerja tetap melalui proses validasi dari Supervisor untuk memastikan kesesuaian dengan kebutuhan manajemen rumah sakit.

3.1.2 Koordinasi

Koordinasi kerja mahasiswa dengan *Supervisor* divisi IT digambarkan melalui alur kerja yang sistematis pada Gambar 3.1, yang memberikan visualisasi tentang bagaimana sebuah tugas dikelola sejak diterima, didiskusikan, dikerjakan, hingga pada akhirnya dievaluasi dan diselesaikan. Alur tersebut dirancang untuk memastikan setiap pekerjaan berjalan sesuai standar perusahaan, serta memudahkan proses pengawasan oleh *Supervisor*. Dengan adanya bagan koordinasi ini, mekanisme kerja menjadi lebih terstruktur, transparan, dan mudah dipahami, baik oleh mahasiswa maupun pihak yang melakukan pengawasan.



Gambar 3. 1 Alur Koordinasi

Pada Gambar 3.1 memperlihatkan bahwa koordinasi dimulai dari penerimaan tugas dari *Supervisor*. Pada tahap ini, mahasiswa menerima arahan awal yang berisi gambaran umum tugas, tujuan pekerjaan, serta tenggat waktu yang harus dipenuhi. Setelah itu dilanjutkan dengan diskusi bersama *Supervisor*, yang berfungsi untuk memperjelas *detail* kebutuhan teknis maupun non-teknis, seperti spesifikasi fitur yang harus dikembangkan, kriteria desain yang harus dipenuhi, atau standar keamanan yang harus diterapkan. Diskusi ini juga menjadi sarana untuk menyamakan persepsi sehingga tidak terjadi perbedaan pemahaman antara pemberi tugas dan pelaksana. Tahap berikutnya adalah pengerjaan tugas berdasarkan hasil

diskusi. Pada fase ini mahasiswa bekerja secara mandiri, mulai dari proses perancangan, implementasi kode, pengujian awal, hingga menghasilkan produk yang siap diperlihatkan. Setelah pekerjaan dianggap selesai, hasil tersebut masuk ke tahap presentasi dan evaluasi bersama *Supervisor*. Pada tahap ini, mahasiswa menjelaskan secara rinci proses pengerjaan, kendala yang dihadapi, serta solusi yang diterapkan. *Supervisor* kemudian meninjau hasil pekerjaan, baik dari sisi teknis maupun kesesuaiannya dengan kebutuhan operasional rumah sakit.

Apabila terdapat kekurangan, hasil pekerjaan akan masuk ke tahap revisi. Pada fase ini *Supervisor* memberikan catatan dan masukan yang lebih spesifik, misalnya perbaikan pada desain antarmuka, optimalisasi performa kode, atau penyesuaian fitur agar sesuai dengan standar keamanan. Mahasiswa kemudian melakukan perbaikan sesuai arahan dan kembali menyerahkan hasil revisi. Proses revisi ini bisa berlangsung berulang hingga Supervisor menyatakan pekerjaan memenuhi standar yang ditetapkan. Melalui mekanisme koordinasi seperti yang digambarkan dalam Gambar 3.1, setiap pekerjaan memiliki siklus yang jelas, mulai dari pemberian instruksi, pelaksanaan, evaluasi, hingga revisi dan finalisasi. Alur ini menunjukkan bahwa meskipun mahasiswa bekerja secara individual tanpa tim khusus, pengawasan dan validasi tetap dilakukan secara ketat oleh *Supervisor*.

3.2 Tugas dan Uraian Kerja Magang

Selama periode magang yang berlangsung selama empat bulan, mahasiswa diberikan berbagai tugas yang berkaitan dengan perancangan dan pengembangan *website* Rumah Sakit Mentari. Seluruh tugas ini disusun dan dilaksanakan secara bertahap mengikuti alur waktu yang tercantum dalam Tabel 1.1 *timeline* pelaksanaan magang, sehingga setiap tahap pekerjaan memiliki target yang jelas dan terukur. Rangkaian pekerjaan tersebut meliputi proses awal perancangan hingga tahap akhir implementasi dan *deployment* sistem. Untuk memberikan gambaran yang lebih detail mengenai jenis pekerjaan dan lingkup tanggung jawab yang dilaksanakan, uraian lengkap dapat dilihat pada Tabel 3.1 detail pekerjaan yang dilakukan.

Table 3.1 Detail Pekerjaan yang Dilakukan

No.	Aktifitas	Minggu ke-	Tanggal Mulai Aktifitas	Tanggal Akhir Aktifitas
1	**Membuat Design Antarmuka**	**1-2 September**	**1 September**	**15 September**
1.1	Perancangan desain UI/UX serta pembuatan *wireframe* sebagai acuan tampilan *website*	1-2 September	1 September	10 September
1.2	Diskusi vengenai design yang akan di implementasikan ke dalam *website*	2 September	11 September	12 September
2	**Membuat Frontend Website Rumah Sakit Mentari**	**2 September - 4 Oktober**	**15 September**	**31 Oktober**
2.1	Pembuatan struktur layout utama dan pengaturan gaya visual *website* menggunakan Tailwind CSS	2 September - 3 Oktober	15 September	14 Oktober
2.2	Implementasi tampilan responsif di berbagai perangkat	3-4 Oktober	15 Oktober	31 Oktober
3	**Membuat Backend Website Rumah Sakit Mentari**	**1 November - 1 Desember**	**3 November**	**28 November**
3.1	Pembuatan *database* (implementasi MySQL/PostgreSQL)	1 November	3 November	7 November
3.2	Melakukan pembuatan backend (API, autentikasi, integrasi *database*)	2-3 November	10 November	18 November
4	**Melakukan Optimasi & Maintenance**	**2-4 Desember**	**5 Desember**	**31 Desember**
4.1	Melakukan optimasi navigasi, hierarki, dan performa *website* (termasuk implementasi CDN)	2 Desember	8 Desember	12 Desember
4.2	Implementasi content Melakukan debugging, troubleshooting, dan optimasi *database*	3 Desember	15 Desember	19 Desember
4.3	Melakukan implementasi keamanan serta testing dan *deployment* ke server	4 Desember	22 Desember	30 Desember

Dalam mendukung proses perancangan dan implementasi *website* Pelayanan Rumah Sakit Mentari, terdapat beberapa *tools* yang digunakan untuk menyelesaikan tugas-tugas yang ada pada Tabel 3.1. *Tools* yang digunakan memiliki kesesuaian berdasarkan fungsional dan kebutuhan teknis dalam pengembangan sistem yang digunakan dalam proyek ini.

1. Visual Studio Code (VS Code)

   Visual Studio Code adalah *source-code editor* ringan namun kuat yang dikembangkan oleh Microsoft dan berjalan di berbagai sistem operasi. Dalam proyek ini, VS Code digunakan sebagai alat utama untuk penulisan, pengeditan, dan debugging kode. VS Code dipilih karena dukungannya yang luas terhadap berbagai bahasa pemrograman (terutama JavaScript, TypeScript, HTML, dan CSS yang relevan untuk *Next.js*), integrasi Git yang mulus, serta ekosistem ekstensi yang kaya untuk membantu mempercepat alur kerja pengembangan front-end dan back-end.

2. MySQL

   MySQL adalah sebuah *Relational Database Management System* (RDBMS) open-source yang populer dan andal. Pada perancangan dan implementasi *website* Pelayanan Rumah Sakit Mentari, MySQL dimanfaatkan sebagai sistem basis data utama. Fungsinya adalah untuk menyimpan, mengelola, dan mengambil semua data terstruktur yang terkait dengan operasional *website*, seperti data pengguna, informasi layanan, jadwal, dan data relasional penting lainnya. MySQL dipilih karena performanya yang cepat, skalabilitas, dan kemudahan integrasinya dengan teknologi back-end yang digunakan.

Dalam pelaksanaan kerja magang kedua *tools* utama yang digunakan adalah Visual Studio Code (VS Code) dan MySQL, yang secara langsung mendukung seluruh proses pengembangan *website* Rumah Sakit Mentari. VS Code berperan sebagai lingkungan kerja utama untuk menulis, mengedit, dan melakukan *debugging* kode, baik pada tahap pembuatan desain antarmuka, implementasi *frontend* menggunakan Next.js dan *Tailwind* CSS, maupun pada pengembangan

backend seperti pembuatan API dan pengaturan logika sistem. Sementara itu, MySQL digunakan sebagai basis data yang menyimpan seluruh informasi dinamis rumah sakit, mencakup data dokter, layanan, jadwal, artikel, serta konten lain yang dikelola melalui panel admin. MySQL juga menjadi komponen penting dalam integrasi *backend* karena setiap proses pengambilan dan penyimpanan data berjalan melalui sistem ini. Dengan memadukan kedua *tools* tersebut, seluruh rangkaian pekerjaan mulai dari perancangan, implementasi, hingga pengujian dapat dilakukan secara terstruktur, efisien, dan sesuai kebutuhan operasional proyek.

3.2.1 Metodologi Pengembangan Sistem (Prototyping)

Metodologi pengembangan sistem yang digunakan dalam kegiatan kerja magang ini adalah metodologi Prototyping. Metode ini dipilih karena mampu memberikan pendekatan pengembangan sistem yang iteratif, fleksibel, dan berorientasi pada kebutuhan pengguna, sehingga sangat sesuai dengan karakteristik sistem informasi rumah sakit yang dinamis dan melibatkan banyak pemangku kepentingan. Prototyping memungkinkan pengembang untuk menghasilkan model awal sistem dalam waktu relatif singkat, kemudian melakukan evaluasi dan penyempurnaan secara berulang berdasarkan umpan balik dari stakeholder terkait. Dalam pengembangan website pelayanan Rumah Sakit Mentari, metodologi *Prototyping* digunakan untuk memastikan bahwa fitur, tampilan antarmuka, serta alur navigasi website benar-benar sesuai dengan kebutuhan operasional rumah sakit dan ekspektasi pengguna, baik dari pihak internal seperti staf administrasi dan divisi marketing, maupun pengguna eksternal yaitu pasien dan masyarakat umum. Pendekatan ini dinilai lebih efektif dibandingkan metode pengembangan linear karena perubahan kebutuhan dapat segera diakomodasi tanpa harus mengulang proses pengembangan dari awal. Secara umum, tahapan metodologi Prototyping yang diterapkan dalam proyek ini terdiri dari beberapa langkah utama.

1. Analisis Kebutuhan (Requirement Analysis)

   Tahap analisis kebutuhan merupakan tahap awal dan paling krusial dalam metodologi Prototyping. Pada tahap ini dilakukan proses pengumpulan kebutuhan sistem melalui observasi, diskusi, dan wawancara dengan

stakeholder terkait, baik dari pihak internal Rumah Sakit Mentari seperti staf administrasi, divisi marketing, maupun manajemen, serta mempertimbangkan kebutuhan pengguna eksternal yaitu pasien dan masyarakat umum. Analisis difokuskan pada identifikasi permasalahan website lama, seperti keterbatasan responsivitas, kesulitan pembaruan konten, performa yang lambat, serta tidak tersedianya alur layanan digital yang efektif. Selain kebutuhan fungsional seperti penyajian informasi layanan, jadwal dokter, dan sistem appointment, tahap ini juga mencakup kebutuhan nonfungsional seperti keamanan data, kemudahan penggunaan, skalabilitas sistem, dan kompatibilitas pada berbagai perangkat. Hasil dari tahap ini menjadi dasar utama dalam perancangan prototipe sistem.

2. Perancangan Prototipe Awal

Setelah kebutuhan sistem teridentifikasi, tahap selanjutnya adalah perancangan prototipe awal. Pada tahap ini, kebutuhan yang telah dianalisis diterjemahkan ke dalam bentuk rancangan antarmuka dan alur sistem menggunakan wireframe dan mockup. Prototipe dirancang untuk merepresentasikan struktur halaman, tata letak elemen, hierarki navigasi, serta interaksi pengguna dengan sistem. Prototipe ini tidak berfokus pada implementasi teknis secara penuh, melainkan pada visualisasi dan pengalaman pengguna (UI/UX) agar stakeholder dapat memperoleh gambaran nyata mengenai sistem yang akan dikembangkan. Perancangan prototipe dilakukan secara iteratif dengan mempertimbangkan standar desain modern, kemudahan akses informasi, dan konsistensi tampilan di berbagai perangkat.

3. Evaluasi Prototipe oleh Stakeholder

Prototipe yang telah dibuat kemudian dievaluasi oleh stakeholder melalui proses peninjauan dan diskusi. Pada tahap ini, stakeholder memberikan masukan terkait kesesuaian desain dengan kebutuhan operasional rumah sakit, kejelasan alur navigasi, kemudahan penggunaan, serta kelengkapan fitur yang ditawarkan. Evaluasi ini bertujuan untuk

mengidentifikasi ketidaksesuaian antara rancangan prototipe dengan kebutuhan pengguna sebelum sistem dikembangkan secara teknis. Melalui keterlibatan langsung stakeholder, potensi kesalahan desain dapat diminimalkan sejak tahap awal, sehingga mengurangi risiko perubahan besar pada tahap implementasi.

4. Penyempurnaan dan Iterasi Prototipe

Berdasarkan hasil evaluasi, prototipe disempurnakan dengan menyesuaikan masukan dan kebutuhan yang belum terpenuhi. Tahap ini merupakan ciri utama metodologi Prototyping karena proses perbaikan dapat dilakukan secara berulang hingga prototipe dianggap representatif dan disepakati bersama. Penyempurnaan dapat meliputi perubahan tata letak antarmuka, penambahan atau pengurangan fitur, serta penyederhanaan alur penggunaan agar lebih intuitif. Proses iterasi ini memastikan bahwa prototipe yang dihasilkan benar-benar mencerminkan kebutuhan pengguna dan siap dijadikan acuan dalam pengembangan sistem secara penuh.

5. Implementasi Sistem

Setelah prototipe disetujui, tahap selanjutnya adalah implementasi sistem berdasarkan prototipe yang telah disempurnakan. Pada tahap ini, rancangan antarmuka dan alur sistem diterapkan ke dalam bentuk sistem nyata menggunakan teknologi yang telah ditentukan. Implementasi mencakup pengembangan frontend, backend, serta integrasi dengan database dan sistem pendukung lainnya. Seluruh proses implementasi mengacu pada prototipe sebagai pedoman utama agar hasil sistem tetap konsisten dengan kebutuhan pengguna dan desain yang telah disepakati.

6. Pengujian dan Evaluasi Akhir

Tahap terakhir dalam metodologi Prototyping adalah pengujian dan evaluasi akhir. Pengujian dilakukan untuk memastikan bahwa seluruh fungsi sistem berjalan sesuai dengan kebutuhan dan tidak terdapat kesalahan yang mengganggu operasional. Pengujian meliputi uji fungsionalitas, uji

responsivitas pada berbagai perangkat, serta uji keamanan sistem. Setelah pengujian selesai, sistem kembali dievaluasi bersama stakeholder untuk memastikan bahwa website telah memenuhi tujuan pengembangan dan siap digunakan secara operasional. Apabila masih ditemukan kekurangan, dilakukan perbaikan hingga sistem dinyatakan layak untuk diimplementasikan.

3.2.2 Perancangan Sistem

Bagian perancangan sistem pada proyek pengembangan *website* pelayanan Rumah Sakit Mentari berfokus pada pemodelan kebutuhan fungsional dan alur kerja sistem sebelum diimplementasikan ke dalam kode program. Pemodelan ini dilakukan untuk memastikan bahwa seluruh interaksi antara pengguna, admin, dan komponen teknis seperti *database* maupun integrasi *WhatsApp* sudah tergambar dengan jelas sehingga meminimalkan kesalahan pada tahap implementasi. Tiga artefak utama yang disusun pada tahap ini yaitu *use case* diagram*, activity* diagram, dan *Entity Relationship* Diagram (ERD), yang masing-masing merepresentasikan sudut pandang berbeda dari sistem namun saling melengkapi satu sama lain.

1. *Use case* Diagram

   Use case diagram pada Gambar 3.2 menggambarkan hubungan antara aktor dengan fungsionalitas yang disediakan oleh sistem informasi berbasis *website* Rumah Sakit Mentari. Pada Gambar 3.2 ditunjukkan dua aktor utama, yaitu User (pengunjung/pasien) dan Admin, yang masing-masing berinteraksi dengan beberapa *use case* sesuai perannya. User dapat melakukan aksi seperti melihat informasi rumah sakit, melihat dokter, layanan, fasilitas, artikel, testimoni, hingga mengakses fitur hubungi rumah sakit, sedangkan Admin dapat mengelola konten dinamis seperti data dokter, jadwal, layanan, artikel, testimoni, serta informasi kontak agar informasi pada *website* selalu mutakhir.

Gambar 3.2 Use Case Diagram

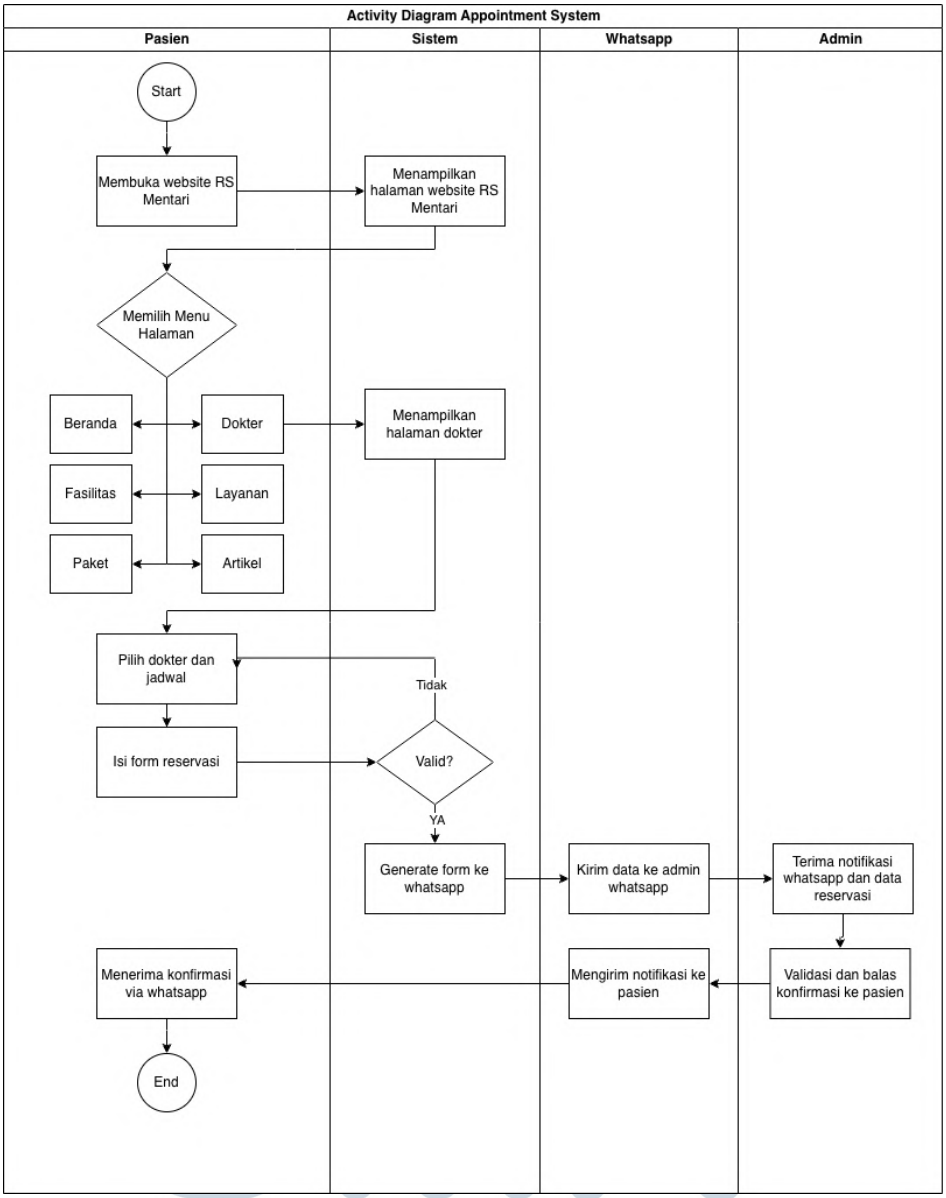
Gambar 3.2 juga menjelaskan batasan sistem (*system boundary*) yang memisahkan fungsionalitas internal *website* dengan aktor eksternal yang berinteraksi dengannya. Dengan memanfaatkan *use case* diagram pada Gambar 3.2, pengembang dan pihak rumah sakit dapat memahami secara visual kebutuhan fungsional apa saja yang wajib didukung oleh sistem sehingga memudahkan proses penelusuran kebutuhan ketika masuk ke tahap perancangan yang lebih detail, seperti perancangan activity diagram maupun struktur basis data. Selain itu, Gambar 3.2 membantu memastikan bahwa tidak ada layanan penting yang terlewat dari sudut pandang pengguna maupun admin.

2. *Activity* Diagram

Activity diagram pada Gambar 3.3 digunakan untuk memodelkan secara rinci alur aktivitas pada proses *appointment* melalui *website* Rumah Sakit Mentari, mulai dari pasien membuka halaman utama sampai menerima konfirmasi melalui *WhatsApp*. Alur dimulai dari aktivitas Pasien dengan Membuka *website* RS Mentari, kemudian dilanjutkan dengan Memilih Menu Halaman yang terdiri dari beberapa pilihan seperti Beranda, Dokter, Fasilitas, Layanan, Paket, dan Artikel sesuai kebutuhan informasi pasien. Ketika pasien memilih menu Dokter, alur pada Gambar 3.3 berlanjut ke aktivitas sistem Menampilkan halaman dokter, kemudian pasien melakukan aksi Pilih dokter

dan jadwal yang sesuai, sebelum akhirnya diarahkan untuk mengisi *form* reservasi sebagai dasar permintaan janji temu. Data yang diisikan pasien tersebut kemudian divalidasi oleh sistem melalui keputusan "Valid?" sehingga hanya data yang lengkap dan benar yang akan diproses lebih lanjut menuju tahap pengiriman ke *WhatsApp*. Dengan adanya pemodelan rinci ini, pengembang dapat lebih mudah mengidentifikasi setiap langkah yang dilalui pengguna dan respon sistem yang harus disediakan. Selain itu, *use case* diagram juga membantu dalam merancang antarmuka dan pesan kesalahan yang jelas agar pengguna terbimbing ketika terjadi kesalahan input.

Gambar 3.3 menunjukkan integrasi antara *website* dan *WhatsApp* dalam proses pengiriman permintaan reservasi ke pihak admin rumah sakit, sehingga menjadi lanjutan langsung dari alur pengisian dan validasi *form* pada *activity* diagram sebelumnya. Setelah data pada *form* reservasi dinyatakan valid sebagaimana digambarkan pada *activity* diagram, sistem akan membentuk pesan terstruktur berisi informasi pasien, dokter, layanan, dan jadwal, kemudian meneruskannya ke *WhatsApp* agar dapat diterima oleh admin untuk ditinjau. Pada Gambar 3.3 juga diperlihatkan bagaimana admin merespons pesan tersebut dengan melakukan pengecekan ketersediaan jadwal lalu mengirimkan konfirmasi atau informasi lanjutan kepada pasien melalui *WhatsApp*. Melalui *activity* diagram, alur proses dari awal sampai akhir dapat dianalisis secara menyeluruh, sehingga potensi kendala seperti kegagalan validasi, keterlambatan konfirmasi, atau kesalahan pengisian data dapat diidentifikasi dan diminimalkan sejak tahap perancangan sebelum sistem benar-benar diimplementasikan.

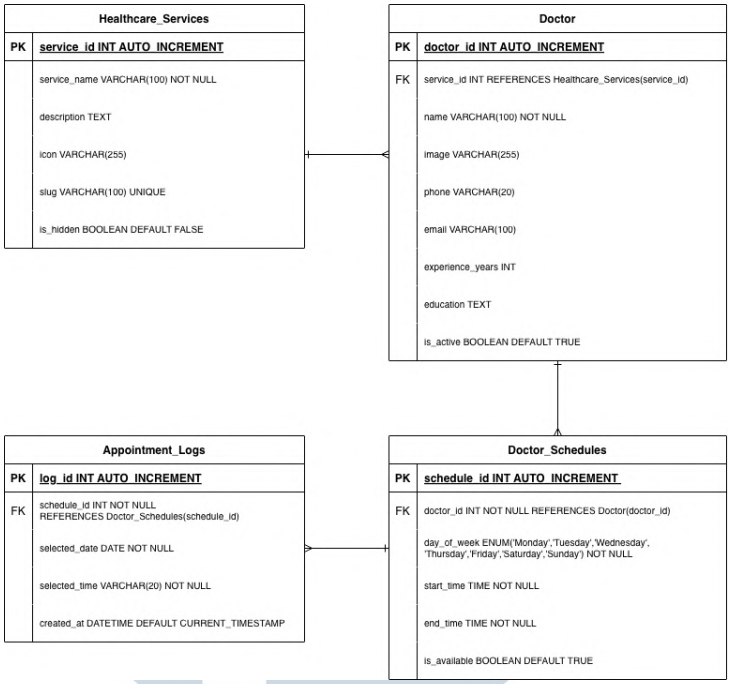


Gambar 3.3 *Activity* Diagram

3. *Entity Relationship* Diagram (ERD)

Entity Relationship Diagram (ERD) pada Gambar 3.4 digunakan untuk merancang struktur basis data yang akan mendukung seluruh proses bisnis pada *website* Rumah Sakit Mentari. Entitas-entitas utama seperti tabel dokter, layanan kesehatan, jadwal dokter, dan data reservasi, beserta atribut penting yang dibutuhkan untuk menyimpan informasi secara terstruktur. Setiap entitas pada Gambar 3.4 dirancang agar dapat merepresentasikan objek dunia

nyata di lingkungan rumah sakit, seperti profil dokter, jenis layanan medis yang tersedia, serta jadwal praktik yang dapat dipilih pasien.



Gambar 3.4 *Entity Relationship* Diagram

Relasi antar entitas pada Gambar 3.4 juga digambarkan dengan jelas untuk menunjukkan hubungan satu-ke-banyak antara layanan dengan dokter, dokter dengan jadwal, serta jadwal dengan data reservasi yang dilakukan pasien. Dengan adanya perancangan ERD, proses normalisasi data dapat dicapai sehingga mengurangi redundansi, mempermudah pemeliharaan, serta meningkatkan efisiensi dalam proses pencarian data ketika *website* diakses. Selain itu ERD berfungsi sebagai acuan utama pada saat pembuatan tabel di *database* server, sehingga implementasi *backend* dapat mengikuti struktur yang sudah terdefinisi dengan baik pada tahap perancangan.

3.2.3 Membuat *Design* Antar Muka

Perancangan desain UI/UX serta pembuatan *wireframe website* Rumah Sakit Mentari dilakukan dengan pendekatan sistematis yang mencakup keseluruhan alur pengembangan, mulai dari tahap konseptual hingga tahap implementasi. Proses dimulai dengan analisis kebutuhan pengguna, yang menjadi dasar utama dalam

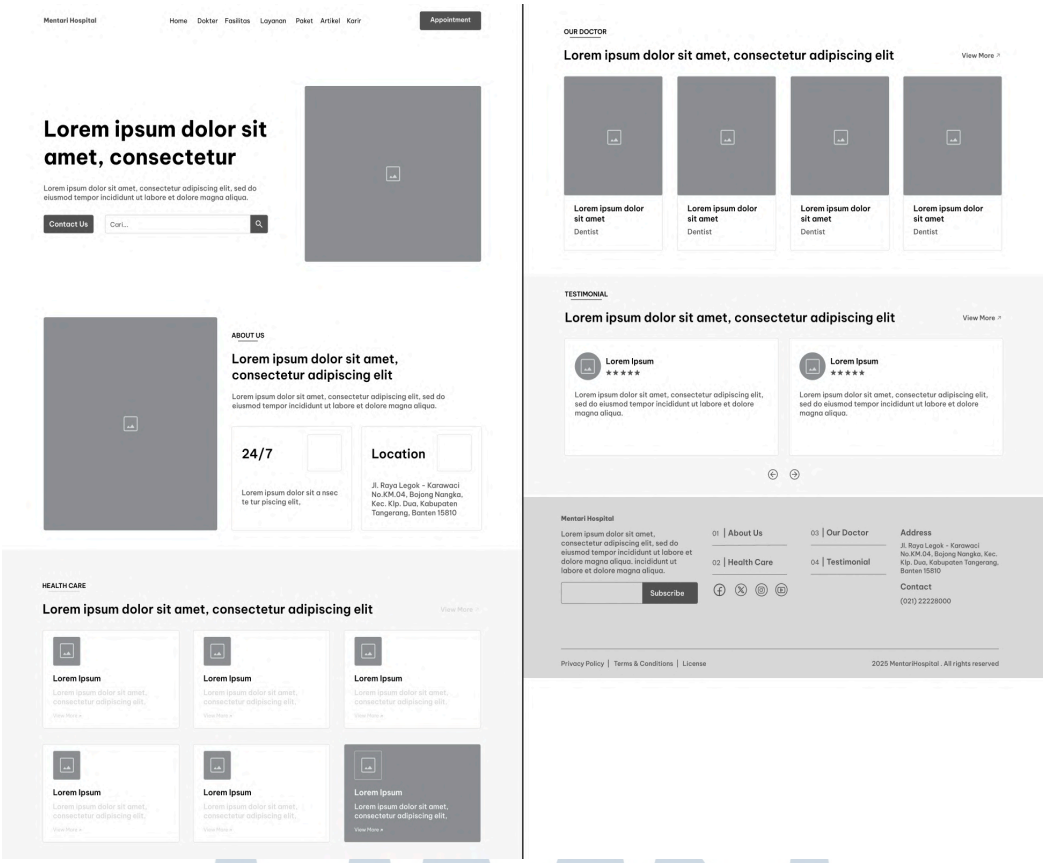
menentukan struktur informasi dan alur navigasi *website*. Analisis ini berfokus pada aspek fungsionalitas utama, seperti akses cepat terhadap jadwal dokter, ketersediaan layanan darurat, informasi fasilitas rumah sakit, serta kemudahan pendaftaran pasien secara daring. Kebutuhan tersebut kemudian dipetakan ke dalam arsitektur informasi yang menggambarkan hubungan antar halaman dan menekankan efisiensi akses, di mana setiap pengguna dapat mencapai informasi yang diinginkan dengan langkah minimal.

3.2.3.1 Perancangan design UI/UX serta pembuatan, wireframe sebagai acuan website

Berdasarkan arsitektur informasi yang telah disusun, proses akan berlanjut ke tahap perancangan *wireframe* menggunakan *Figma*. *Wireframe* berfungsi sebagai kerangka visual awal yang menampilkan tata letak komponen, distribusi konten, serta hierarki informasi, tanpa melibatkan detail visual akhir. Dalam rancangan ini, setiap elemen penting ditempatkan secara strategis untuk mendukung pengalaman pengguna. Bagian *header*, misalnya, memuat logo serta menu navigasi utama seperti beranda, layanan, dokter, fasilitas, dan artikel. *Hero section* dirancang untuk menampilkan pesan utama rumah sakit dengan judul besar, teks pendukung, tombol interaksi seperti *Contact Us* dan *Appointment*, serta area visual berupa gambar yang dapat diganti sesuai kebutuhan promosi.

Di bawah *hero section*, *wireframe* memperlihatkan blok informasi "*About Us*" yang berisi deskripsi singkat rumah sakit, dilengkapi dengan informasi penting seperti layanan darurat 24/7 dan lokasi rumah sakit. Bagian berikutnya adalah "*Our Doctor*," yang menampilkan daftar tenaga medis dalam bentuk kartu sehingga memudahkan pengguna mengenali dokter beserta spesialisasinya. Untuk memperkuat kepercayaan publik, *wireframe* juga menyertakan *section* testimoni pasien yang menampilkan umpan balik dalam format kartu ulasan. Pada bagian akhir halaman, *footer* dirancang berisi informasi kontak, alamat, serta tautan ke kebijakan privasi, syarat dan ketentuan, serta informasi hak cipta. Secara keseluruhan, *wireframe* ini menggambarkan rancangan hierarkis yang logis dan

menjadi media komunikasi visual dengan pemangku kepentingan. *Wireframe* tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.5 *Wireframe* Rumah Sakit Mentari

Proses perancangan pada Gambar 3.2 tidak hanya berhenti pada kerangka awal, tetapi juga mempertimbangkan aspek teknis sejak dini. *Grid layout* digunakan untuk menjaga konsistensi tata letak, sementara prinsip *white space* diterapkan untuk meningkatkan keterbacaan konten. Tipografi ditentukan dengan menyesuaikan tingkat hierarki, di mana teks judul dibuat lebih besar dan kontras dibandingkan dengan teks deskriptif. Setiap komponen utama juga diberi anotasi fungsional, seperti cara kerja tombol, validasi pada formulir pendaftaran, hingga perilaku interaktif seperti *hover state* dan *focus state*. Selain itu, prinsip responsivitas sudah diterapkan sejak tahap *wireframe*, dengan merancang adaptasi tampilan untuk desktop, tablet, dan ponsel pintar, sehingga pengguna tetap mendapatkan pengalaman optimal di berbagai perangkat.

Setelah *wireframe* selesai, tahap berikutnya adalah pembuatan desain antarmuka berfidelitas tinggi (*high-fidelity UI*). Pada tahap ini, detail visual seperti pemilihan palet warna, tipografi final, ikonografi, serta ilustrasi mulai diterapkan. Palet warna dipilih untuk mencerminkan identitas profesional rumah sakit, dengan dominasi warna oranye dan hijau yang selaras dengan logo institusi. Komponen antarmuka, seperti tombol dan *form input*, didesain mengikuti prinsip *design system* agar konsisten di seluruh halaman. Desain interaktif juga diuji langsung dalam *Figma* dengan pembuatan *prototipe* yang memungkinkan simulasi navigasi antar halaman dan interaksi sederhana, sehingga memberikan gambaran nyata pengalaman pengguna.

Tahap akhir dari proses ini adalah implementasi desain ke dalam bentuk *website* fungsional. Seluruh elemen yang telah dirancang pada *Figma* kemudian diterjemahkan ke dalam kode dengan memanfaatkan teknologi modern, yaitu *Next.js* sebagai *framework* utama untuk pengembangan *frontend* dengan dukungan *server-side rendering, Tailwind CSS* sebagai kerangka kerja *styling* yang memastikan konsistensi visual dan efisiensi mahasiswa magangan kode, serta integrasi *database* untuk mendukung fitur dinamis seperti pendaftaran *online* dan manajemen konten. Proses implementasi dilakukan secara menyeluruh, mencakup konversi *layout* ke dalam komponen yang dapat digunakan kembali, pengujian *responsivitas* pada berbagai perangkat, serta penerapan praktik keamanan dasar pada sisi *frontend*.

Dengan adanya alur lengkap mulai dari analisis kebutuhan, penyusunan arsitektur informasi, pembuatan *wireframe*, perancangan *high-fidelity UI*, hingga tahap implementasi, seluruh tahapan pengembangan *website* dapat dilaksanakan secara terpadu dan terstruktur. Pendekatan ini memastikan bahwa hasil akhir *website* tidak hanya menampilkan desain yang modern dan estetis, tetapi juga fungsional, responsif, serta sesuai dengan kebutuhan nyata pengguna.

3.2.4 Membuat *Frontend Website* Rumah Sakim Mentari

Frontend Website Rumah Sakit Mentari dibangun untuk memenuhi kebutuhan informasi dan pelayanan kesehatan yang cepat, mudah diakses, serta ramah pengguna. *Website* rumah sakit memiliki karakteristik khusus, yaitu harus dapat memberikan informasi yang akurat, mudah ditemukan oleh mesin pencari, serta tetap ringan ketika diakses oleh berbagai perangkat baik desktop maupun *mobile*. Untuk itu, arsitektur *frontend* dirancang menggunakan kombinasi *framework*, *library*, dan sistem *styling modern*, yaitu *Next.js* sebagai *framework* utama, *React* sebagai *library* UI, dan *Tailwind CSS* sebagai sistem styling. Ketiganya dipilih karena mendukung prinsip modularitas, efisiensi, serta kemudahan dalam pengembangan berkelanjutan.

Arsitektur *frontend* menggunakan pendekatan component-based architecture. Semua fitur utama *website*, seperti *Navbar*, *About Us, HealthCare, Footer, OurDoctor*, Testimonial, Rawat Inap, dan *Page Cover*, dipisahkan ke dalam folder masing-masing. Setiap folder berisi file *index.tsx* sebagai *entry point* serta subfolder *source* untuk aset seperti gambar dan ikon. Pola ini memberikan struktur yang rapi, mudah dipahami, serta mempermudah proses pemeliharaan dan pengembangan lebih lanjut. Selain itu, data yang digunakan pada *website* seperti daftar layanan kesehatan, daftar dokter, serta spesialisasi dikelola secara terpusat melalui berkas data. Data tersebut kemudian dialirkan ke komponen dengan pola *unidirectional data flow* agar konsistensi tetap terjaga. Untuk kebutuhan administratif yang memerlukan pengelolaan konten, diterapkan state management sederhana dengan memanfaatkan library ringan sehingga perubahan dapat ditampilkan secara reaktif tanpa mempengaruhi performa aplikasi secara keseluruhan.

Routing mengikuti sistem *file-based routing* dari *Next.js*, sehingga pembuatan halaman statis, halaman dinamis dengan parameter slug, hingga *nested routing* dapat dilakukan dengan lebih mudah dan terstruktur. Navigasi dibuat *responsive*, menu horizontal digunakan pada tampilan desktop, sementara untuk perangkat *mobile* digunakan menu hamburger dengan *dropdown* vertikal. Untuk meningkatkan pengalaman pengguna, link aktif diberi tanda visual berupa

underline berwarna oranye sehingga pengguna mengetahui posisi mereka di *website*.

1. Framework *Next.js*

Next.js menjadi kerangka utama karena mendukung *Server-Side Rendering* (SSR) dan *Static Site Generation* (SSG). Dengan SSR, halaman dapat di-render di server sebelum dikirim ke browser, sehingga mempercepat waktu tampil (time to first byte) dan memastikan konten penting langsung tersedia tanpa harus menunggu proses JavaScript di sisi klien. Sementara SSG digunakan untuk halaman-halaman statis seperti profil rumah sakit, daftar layanan, dan artikel kesehatan, yang jarang berubah sehingga dapat dipre-render untuk meningkatkan kecepatan akses. *Next.js* juga memiliki sistem optimisasi gambar bawaan yang mendukung *lazy loading* dan *responsive sizing*, sehingga gambar ditampilkan sesuai kebutuhan perangkat pengguna tanpa mengurangi kualitas. *Routing* berbasis *file* memberikan kemudahan dalam mengatur struktur halaman, sementara fitur *automatic code splitting* memecah bundle aplikasi agar hanya kode yang diperlukan pada halaman tertentu yang akan dimuat, mengurangi ukuran *file* yang diunduh pengguna.

Next.js mendukung *prefetching link* sehingga halaman yang akan dikunjungi dapat dimuat lebih cepat karena resource-nya sudah dipersiapkan di background. Fitur ini sangat berguna untuk *website* rumah sakit yang memiliki banyak halaman informasi, sehingga navigasi antarhalaman terasa instan. Dukungan untuk SEO juga lebih optimal karena *Next.js* menyediakan integrasi dengan Head component untuk mengatur *meta tag, title*, dan deskripsi yang sesuai dengan standar mesin pencari.

2. Library *React*

React digunakan sebagai dasar dalam pembangunan antarmuka berbasis komponen. Dengan sifatnya yang deklaratif, setiap komponen dibangun berdasarkan perubahan *state* dan *props*, sehingga ketika data berubah, tampilan langsung diperbarui secara otomatis tanpa perlu

manipulasi DOM secara manual. Komponen fungsional dan *React Hooks* dimanfaatkan untuk mengelola state, efek samping (side effects), serta logika reaktif pada UI. Contohnya, penggunaan useState pada Navbar memungkinkan toggle menu *mobile* berjalan mulus, sementara *useEffect* digunakan untuk mengatur perilaku tertentu saat state berubah. Struktur berbasis komponen ini memungkinkan elemen-elemen seperti kartu dokter, daftar layanan kesehatan, dan form pendaftaran dibangun sekali lalu digunakan berulang kali di berbagai halaman. Integrasi dengan *TypeScript* memperkuat pengembangan dengan menambahkan tipe data pada props dan state. Hal ini meminimalisasi potensi *bug* saat data berpindah antar komponen dan membuat kode lebih mudah dibaca serta dipelihara dalam jangka panjang. *React* juga mendukung optimisasi performa melalui memoization dan *lazy loading*, sehingga komponen besar dapat dipisahkan menjadi bagian lebih kecil yang hanya dimuat ketika diperlukan.

3. *Tailwind CSS* sebagai Sistem *Styling*

Tailwind CSS digunakan sebagai sistem *styling* utama karena menawarkan pendekatan utility-first yang memungkinkan proses desain dilakukan dengan cepat, konsisten, dan efisien. Dengan metode ini, elemen visual dapat dibentuk secara langsung berdasarkan utilitas yang sudah disediakan oleh *Tailwind*, sehingga tidak diperlukan mahasiswa magangan *stylesheet* kustom yang panjang dan berulang. Hal ini menjadikan proses pengembangan antarmuka lebih ringkas sekaligus terstandarisasi. Dalam penerapannya, *Tailwind CSS* dikonfigurasi untuk menyesuaikan dengan identitas visual Rumah Sakit Mentari. SPenyesuaian tersebut mencakup pemilihan tipografi utama yang konsisten di seluruh halaman serta penetapan warna brand khusus sebagai warna utama. Warna brand ini digunakan secara menyeluruh, baik pada elemen navigasi, tombol, maupun penanda interaksi, sehingga menciptakan kesan visual yang harmonis dan profesional.

Tailwind CSS juga mendukung desain responsif secara menyeluruh, memungkinkan tampilan *website* menyesuaikan dengan berbagai ukuran

layar, mulai dari perangkat *mobile* hingga desktop. Dengan fleksibilitas ini, pengguna tetap mendapatkan pengalaman yang optimal tanpa terganggu oleh perbedaan perangkat. Selain itu, sistem ini menghasilkan file akhir yang lebih ringan karena hanya gaya yang benar-benar digunakan yang akan diproses dan dimasukkan ke dalam proyek. Hasilnya, *website* menjadi lebih cepat diakses, performa tetap terjaga, dan pengalaman pengguna semakin baik. Dukungan pada pengaturan global seperti tipografi, warna, dan tata letak juga memastikan bahwa tampilan *website* tetap konsisten di setiap halaman.

4. Responsivitas, Aksesibilitas, dan Optimasi

Desain *frontend* dirancang *mobile-first*, memastikan tampilan dapat digunakan dengan baik pada perangkat *smartphone* sebelum diperluas ke layar yang lebih besar. Tata letak menggunakan kombinasi *flexbox* dan *grid* dari *Tailwind* untuk memastikan konten dapat beradaptasi pada berbagai ukuran layar. Aksesibilitas diprioritaskan melalui penggunaan struktur *HTML* semantik, teks alternatif pada semua gambar, dukungan navigasi *keyboard*, serta kontras warna yang sesuai standar WCAG. Dengan demikian, *website* dapat digunakan oleh pengguna dengan kebutuhan khusus, termasuk pasien lansia atau pengguna dengan keterbatasan visual. Dari sisi optimasi, diterapkan *lazy loading* pada gambar, *code splitting* otomatis untuk setiap halaman, *caching* aset statis di CDN, serta penggunaan ikon berbasis SVG untuk menjaga ukuran *file* tetap ringan. Selain itu, efek interaktif seperti transisi halus, *hover state* yang jelas, dan *backdrop* blur pada *navbar* diterapkan untuk memberikan pengalaman pengguna yang *modern* namun tetap mudah digunakan.

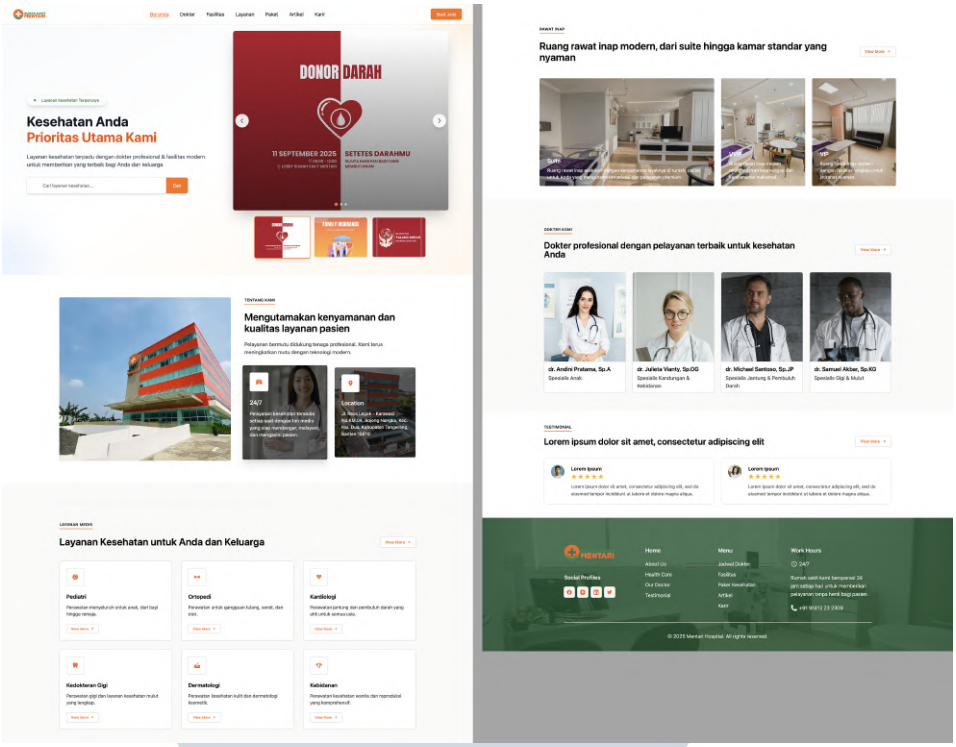
3.2.4.1 Tampilan beranda pembuatan struktur layout utama dan pengaturan gaya visual menggunakan *Tailwind CSS*

Tampilan pada sebuah *website* rumah sakit memiliki peran penting sebagai pintu masuk utama pengguna untuk mengenal layanan, fasilitas, serta kualitas institusi kesehatan tersebut. Tampilan awal yang informatif, terstruktur, dan mudah dipahami akan sangat membantu pasien atau pengunjung baru dalam memahami

apa yang ditawarkan tanpa perlu mencari terlalu jauh. Dalam konteks desain modern, halaman beranda biasanya juga menggabungkan elemen visual seperti foto fasilitas, profil dokter, dan testimoni untuk membangun rasa kepercayaan sejak pertama kali pengguna melihat halaman. Pada gambar 3.5 setiap bagian dalam beranda disusun dengan pertimbangan alur berpikir pengguna, mulai dari pengenalan brand, informasi layanan inti, sampai dorongan untuk melakukan aksi seperti membuat janji. Dengan memahami seluruh komponen ini secara menyeluruh, kita bisa melihat bagaimana beranda bekerja bukan hanya sebatas tampilan, tetapi sebagai pengalaman yang memandu pengguna dari orientasi hingga keputusan

1. Tampilan Beranda

Tampilan beranda pada sebuah website rumah sakit memiliki peran penting sebagai pintu masuk utama bagi pengguna untuk mengenal layanan, fasilitas, serta kualitas institusi kesehatan yang ditawarkan, sehingga penyajian informasi harus dirancang secara informatif, terstruktur, dan mudah dipahami agar pasien maupun pengunjung baru dapat langsung memperoleh gambaran umum tanpa perlu menelusuri banyak halaman. Dalam desain website modern, beranda tidak hanya berisi teks, tetapi juga dilengkapi dengan elemen visual seperti foto fasilitas, profil dokter, ikon layanan, dan testimoni pasien yang berfungsi untuk membangun kepercayaan serta menampilkan citra profesional rumah sakit sejak kesan pertama. Pada Gambar 3.5, setiap bagian beranda disusun berdasarkan alur berpikir pengguna, dimulai dari pengenalan identitas dan brand rumah sakit, penyampaian layanan inti, hingga penyediaan tombol atau fitur aksi seperti pembuatan janji temu yang memudahkan pengguna untuk langsung berinteraksi. Susunan ini bertujuan agar pengguna dapat memahami informasi secara bertahap dan logis tanpa merasa bingung atau terbebani. Dengan demikian, halaman beranda tidak hanya berfungsi sebagai tampilan awal, tetapi juga sebagai sarana pengalaman pengguna yang efektif dalam mengarahkan pengunjung dari tahap pengenalan hingga pengambilan keputusan untuk memanfaatkan layanan rumah sakit.



Gambar 3.6 Tampilan Beranda

Bagian header pada Gambar 3.6 merupakan elemen paling atas pada halaman beranda yang berfungsi sebagai titik awal interaksi pengguna dengan *website*. Pada area ini ditampilkan logo rumah sakit yang ditempatkan di sisi kiri, sehingga mudah dikenali oleh pengguna yang baru pertama kali mengakses halaman. Selain logo, terdapat deretan menu navigasi seperti Beranda, Dokter, Fasilitas, Layanan, Paket, Artikel, dan Karir yang disusun secara horizontal untuk mempermudah proses pencarian informasi. Penempatan menu seperti ini membantu pengguna memahami struktur *website* hanya melalui sekilas pandang, sehingga mereka tidak perlu melakukan banyak eksplorasi. Di sisi kanan *header* terdapat tombol "Buat Janji" yang ditampilkan dalam warna mencolok agar lebih mudah ditemukan oleh pengunjung yang ingin melakukan reservasi layanan. Tombol ini juga berfungsi sebagai *call-to-action* utama yang mendorong pengguna melakukan tindakan cepat tanpa harus membuka halaman lain terlebih dahulu.

Bagian tentang kami pada Gambar 3.6 menampilkan informasi mengenai identitas rumah sakit secara singkat namun jelas agar pengguna langsung memperoleh gambaran umum. Pada bagian ini ditampilkan foto gedung rumah sakit yang memberikan kesan profesional dan menunjukkan fasilitas fisik yang dimiliki. Di samping foto, terdapat paragraf deskripsi yang menjelaskan komitmen rumah sakit dalam memberikan pelayanan terbaik, termasuk dukungan tenaga kesehatan profesional dan penggunaan teknologi modern. Informasi ini bertujuan agar pengguna merasa lebih yakin terhadap kualitas layanan yang ditawarkan. Selain teks utama, terdapat juga beberapa poin penting seperti layanan 24 jam dan lokasi rumah sakit yang ditunjukkan melalui kartu-kartu kecil dengan ikon. Elemen visual ini membantu pengguna menangkap informasi penting tanpa harus membaca panjang lebar.

Bagian layanan kesehatan (*healthcare*) pada Gambar 3.6 menyajikan berbagai jenis layanan medis yang tersedia di rumah sakit dalam bentuk kartu-kartu informasi. Setiap kartu mewakili satu layanan, seperti Pediatri, Ortopedi, Kardiologi, Kedokteran Gigi, Dermatologi, dan Kebidanan, sehingga pengguna dapat melihat pilihan layanan yang ditawarkan secara cepat. Pada masing-masing kartu terdapat judul layanan dan deskripsi singkat untuk membantu pengguna memahami cakupan layanannya. Penataan kartu dalam bentuk *grid* mempermudah navigasi visual, khususnya bagi pengguna yang hanya ingin sekadar melakukan pencarian awal. Setiap kartu juga dilengkapi dengan tombol “View More” untuk mengarahkan pengguna ke halaman informasi yang lebih rinci. Tampilan bagian ini memberikan kesan bahwa rumah sakit memiliki layanan komprehensif yang dapat memenuhi berbagai kebutuhan kesehatan keluarga.

Bagian rawat inap pada Gambar 3.6 menampilkan informasi mengenai pilihan kamar yang tersedia bagi pasien yang membutuhkan perawatan jangka lebih lama. Pada bagian ini ditampilkan beberapa foto ruang rawat inap seperti kamar standar, kamar *VIP*, atau kamar *suite* yang disusun dalam bentuk galeri. Foto-foto tersebut memberikan gambaran nyata mengenai

kondisi kamar, mulai dari kebersihan, tata letak, hingga fasilitas yang tersedia di dalamnya. Setiap tampilan foto dilengkapi dengan penjelasan singkat mengenai tipe kamar dan keunggulan masing-masing, sehingga pasien dapat mempertimbangkan kenyamanan yang dibutuhkan. Penyajian visual ini sangat membantu karena banyak pasien memilih kamar berdasarkan fasilitas fisik yang terlihat langsung. Adanya tombol "View More" memberikan kesempatan bagi pengguna untuk mempelajari detail lebih lanjut sebelum mengambil keputusan.

Bagian dokter kami pada Gambar 3.6 memperkenalkan tenaga medis yang bekerja di rumah sakit sebagai salah satu aspek penting dalam pelayanan kesehatan. Profil setiap dokter ditampilkan dalam bentuk kartu yang berisi foto profesional, nama lengkap, dan gelar akademik untuk menunjukkan kredibilitas. Pada kartu yang sama juga tercantum spesialisasi masing-masing dokter, seperti spesialis jantung, kebidanan, atau penyakit dalam. Penampilan profil dalam format visual seperti ini memudahkan pengguna mengenali dokter yang sesuai dengan kebutuhan mereka. Foto dokter yang ditampilkan turut memberikan kesan ramah dan profesional, sehingga pengguna merasa lebih nyaman sebelum bertemu langsung. Bagian ini membantu meningkatkan rasa kepercayaan karena calon pasien bisa mengetahui kompetensi tenaga medis yang tersedia.

Bagian testimonial pada Gambar 3.6 berisi ulasan singkat dari pasien yang telah menggunakan layanan rumah sakit sebelumnya. Setiap testimonial menampilkan foto kecil pasien (atau avatar), nama, rating bintang, serta komentar singkat mengenai pengalaman layanan yang mereka peroleh. Tampilan testimonial dalam bentuk kartu membuat informasi mudah dibaca dan memberi ruang visual yang cukup antar elemen. Ulasan seperti ini memberikan bukti bahwa pelayanan yang diberikan rumah sakit telah dirasakan positif oleh pengunjung lain. Keberadaan testimonial membantu pengunjung baru dalam mengambil keputusan, terutama bagi mereka yang ragu sebelum menggunakan layanan kesehatan tertentu. Penempatan bagian

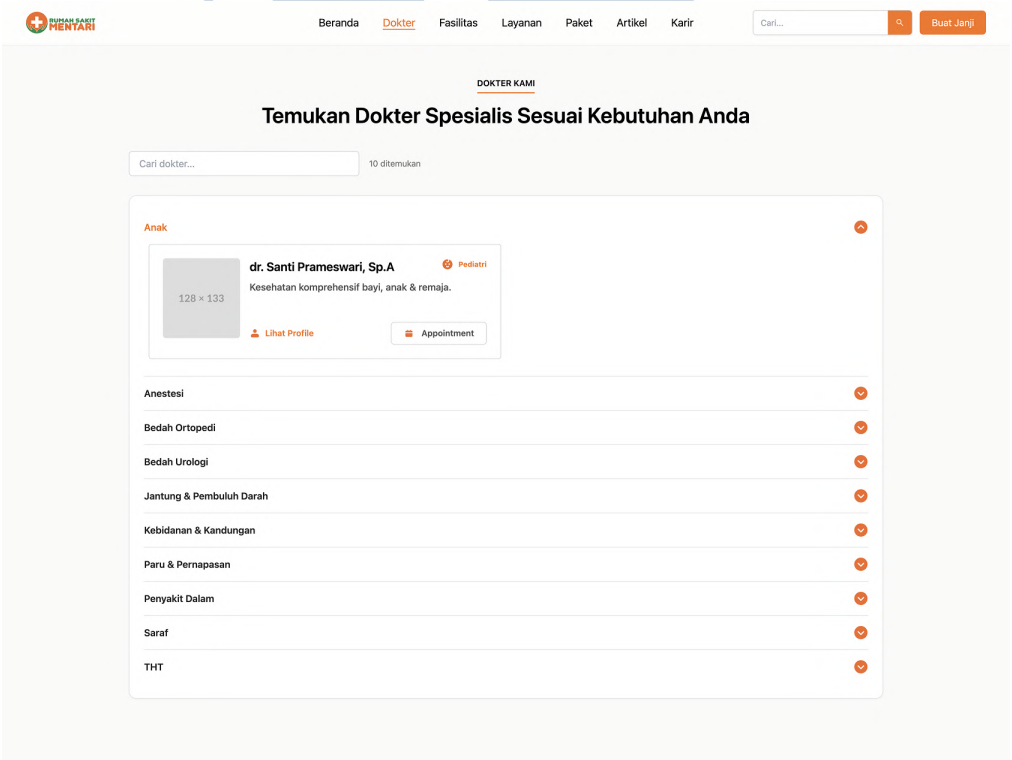
ini juga menunjukkan bahwa rumah sakit terbuka terhadap feedback dan mengutamakan kepuasan pasien.

Footer merupakan bagian paling bawah dari halaman yang berfungsi sebagai pusat informasi tambahan dan akses cepat ke berbagai bagian penting *website*. Pada footer ditampilkan kembali logo rumah sakit sebagai penguat identitas visual. Selain itu, terdapat daftar menu tambahan seperti Tentang Kami, Health Care, Rawat Inap, Dokter Kami, dan Testimonial yang ditata dalam bentuk kolom. Footer juga memuat informasi penting seperti jam operasional rumah sakit, nomor kontak, serta ikon media sosial yang mengarahkan pengguna ke platform resmi rumah sakit. Penempatan alamat dan kontak dalam footer memudahkan pengguna yang ingin mendapatkan informasi praktis tanpa harus kembali ke bagian atas halaman. Desain footer dibuat dengan latar berwarna lebih gelap untuk membedakannya dari konten utama dan menutup halaman dengan tampilan yang rapi.

2. Tampilan Menu Dokter

Tampilan menu Dokter pada *website* rumah sakit, dimulai dengan pengantar singkat lalu diikuti penjelasan lengkap dan terperinci setiap komponennya. Menu Dokter berfungsi sebagai halaman utama bagi pasien untuk mencari, mengenal, dan memilih tenaga medis yang sesuai dengan kebutuhan mereka. Halaman ini dirancang agar informasinya mudah dicari, terpercaya, dan mendukung tindakan selanjutnya seperti melihat jadwal atau membuat janji. Struktur dan bahasa yang digunakan dibuat sederhana agar semua umur bisa memahami, sementara konten klinis tetap tersaji secara profesional. Navigasi di menu Dokter terintegrasi dengan sistem *booking* dan layanan lainnya sehingga pasien tidak perlu berpindah antar halaman terlalu banyak saat ingin melakukan reservasi atau konsultasi. Desainnya juga memperhatikan aksesibilitas dasar supaya pengguna dengan kebutuhan khusus tetap dapat menggunakan fitur pencarian dan pemilihan dokter. Pada menu Dokter Gambar 3.7 menampilkan judul halaman seperti “Dokter Kami” atau “Cari Dokter” beserta jalur navigasi singkat yang menunjukkan

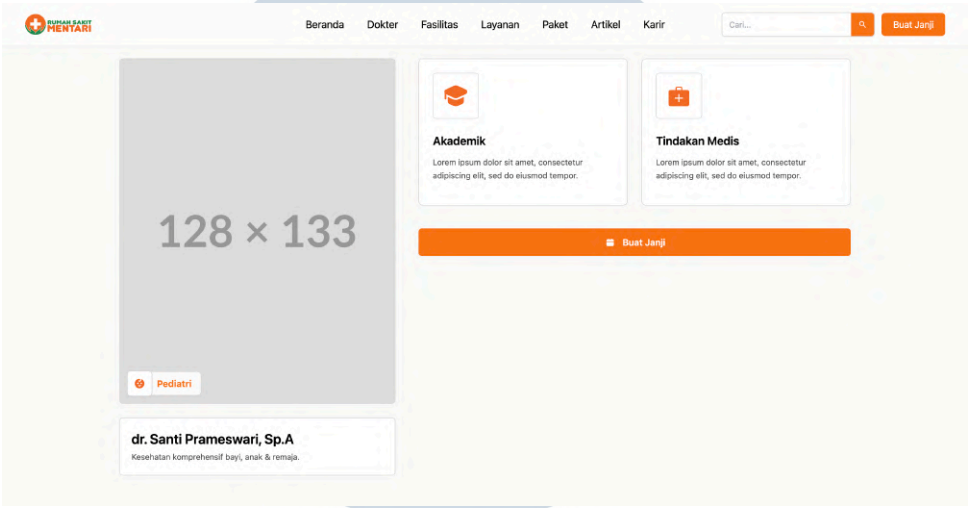
lokasi pengguna di dalam situs. Area *header* ini biasanya memuat ringkasan singkat jumlah dokter terdaftar. Di samping judul dapat disediakan tombol cepat untuk membuat janji baru dan link ke FAQ tentang tata cara konsultasi, sehingga pengguna yang sudah paham proses bisa langsung melanjutkan. Penempatan elemen-elemen ini dibuat konsisten dengan header global sehingga pengguna tidak kehilangan orientasi di dalam *website*. Warna dan tipografi pada header menjaga kontras agar mudah dibaca oleh pengguna.



Gambar 3.7 Tampilan Menu Dokter

Pada Gambar 3.7 fitur pencarian dan filter adalah komponen utama di bagian atas daftar dokter, dirancang untuk membantu pengguna menemukan dokter berdasarkan kriteria spesifik. Pencarian teks memungkinkan pencarian berdasarkan nama dokter, gelar, atau kata kunci spesialisasi seperti "kardiologi" atau "anak", dan hasil langsung ditampilkan saat pengguna mengetik apa bila tersedia. Filter yang umum meliputi spesialisasi dokter dalam bidang tertentu. Untuk pengguna *mobile*, filter disusun dalam *drawer* atau modal agar ruang layar tetap efisien. Daftar dokter ditampilkan dalam

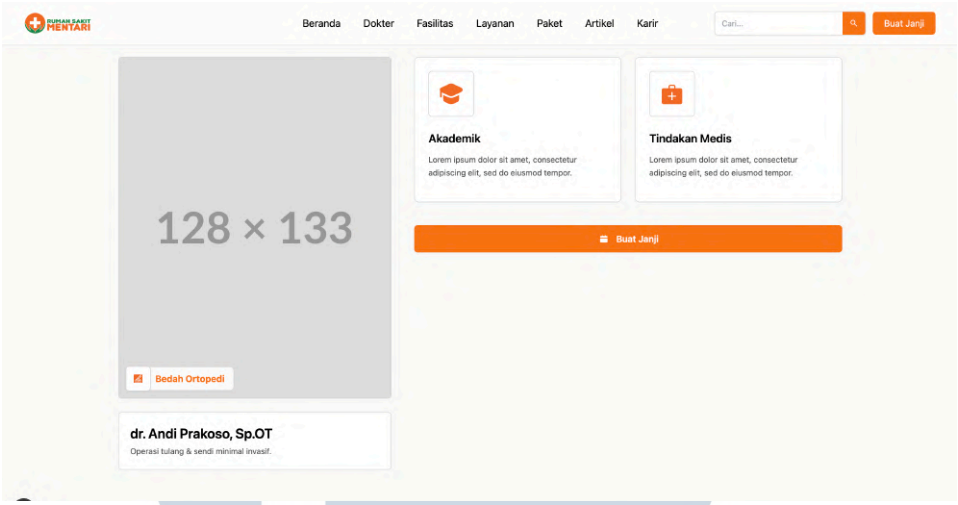
format kartu atau bar list yang konsisten, setiap entri menampilkan foto profesional, nama lengkap dan gelar, spesialisasi utama. Di bagian akhir kartu tersedia dua tombol utama “Lihat Profil” untuk membuka halaman detail dokter dan “Buat Janji” yang membuka modal booking. Tata letak kartu disesuaikan agar informasi penting terlihat tanpa harus membuka detail, serta memberikan visual untuk tindakan lanjut.



Gambar 3.8 Tampilan Menu Dokter

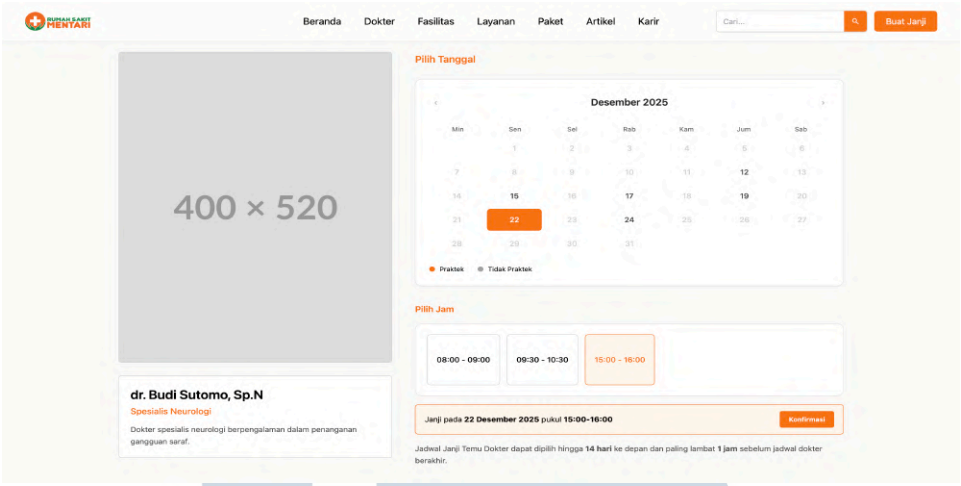
Halaman profil dokter pada Gambar 3.8 menyajikan informasi lengkap dan terstruktur tentang latar belakang profesional setiap dokter sehingga pasien mendapat gambaran mendalam sebelum membuat keputusan. Di bagian profil terdapat foto dokter, nama lengkap, gelar akademik, spesialisasi lengkap, serta afiliasi institusi dan jabatan jika ada. Selanjutnya ada ringkasan singkat yang mencakup pendidikan, pengalaman klinis, dan area kompetensi yang sering ditangani. Bagian jadwal menampilkan hari dan jam praktik secara rinci, dengan label untuk slot yang masih tersedia dan tombol untuk melihat detail slot waktu yang tersedia untuk melakukan buat janji. Informasi yang ditampilkan pada halaman ini dibuat secara ringkas namun tetap komprehensif agar pasien dapat memahami profil dokter tanpa merasa terbebani oleh informasi berlebihan. Selain itu, tampilan yang rapi dan pengelompokan informasi yang jelas membantu meningkatkan kenyamanan pengguna saat menelusuri halaman tersebut. Dengan penyajian data yang

detail namun tetap mudah dipahami, halaman profil dokter mampu menjadi sumber informasi yang kredibel dan memudahkan pasien dalam menentukan dokter yang paling sesuai dengan kebutuhan mereka.



Gambar 3.9 Tampilan Menu Dokter

Halaman Buat Janji dan Manajemen Jadwal pada Gambar 3.9 dirancang dengan pendekatan antarmuka yang intuitif untuk memastikan proses pemesanan berlangsung sederhana dan cepat, namun tetap mampu menghimpun informasi komprehensif yang dibutuhkan oleh sistem administrasi rumah sakit. Saat pengguna menekan tombol “Konfirmasi”, sistem akan menampilkan form bertahap (multi-step form) yang berfungsi memvalidasi data secara sistematis, mulai dari identitas dasar pasien seperti nama lengkap, NIK, tanggal lahir, jenis kelamin, alamat, dan nomor telepon, hingga informasi klinis seperti riwayat kunjungan sebelumnya serta keluhan kesehatan yang dirasakan. Selain itu, form ini memandu pengguna dalam memilih layanan atau klinik spesialis yang tepat serta menyajikan pilihan slot waktu secara real-time berdasarkan jadwal dokter yang tersedia, sehingga meminimalisir risiko kesalahan input dan penumpukan antrean sekaligus meningkatkan efisiensi operasional rumah sakit dalam mengelola data kunjungan pasien.



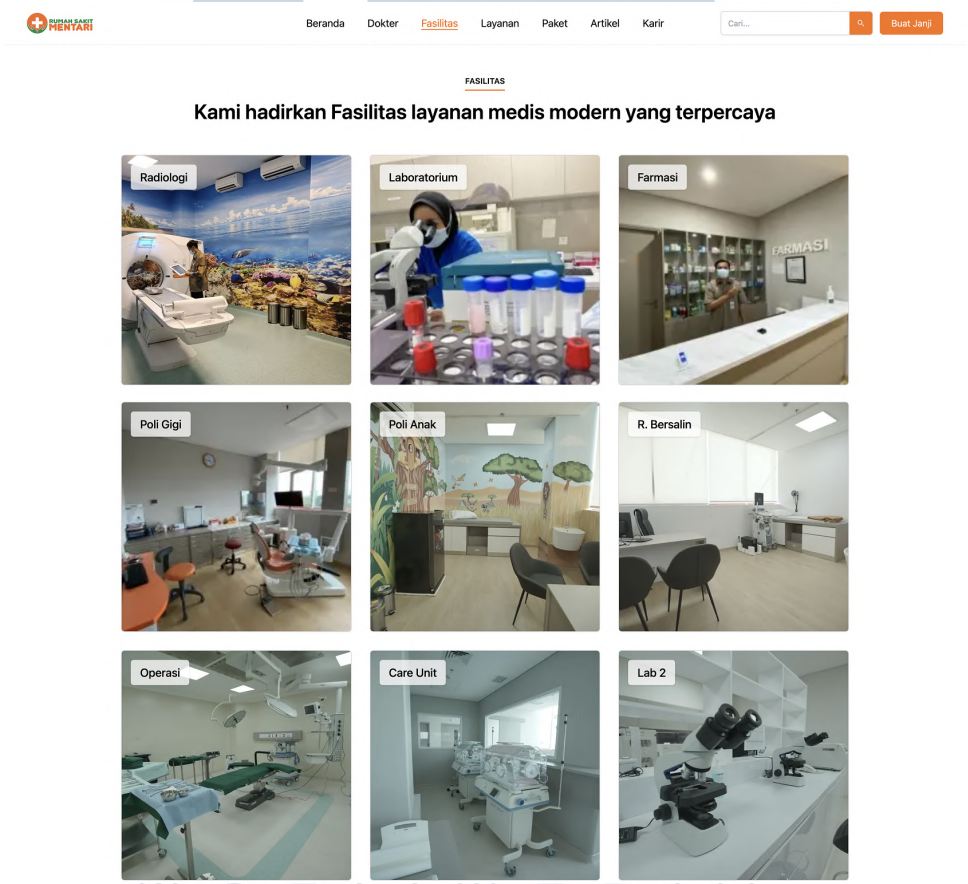
Gambar 3.10 Tampilan Menu Dokter

3. Tampilan Menu Fasilitas

Menu "Fasilitas" di RS Mentari memperkenalkan layanan–layanan dan fasilitas medis yang tersedia di rumah sakit secara transparan pada Gambar 3.10. Di halaman ini, pengunjung bisa melihat daftar fasilitas utama seperti layanan radiologi, laboratorium, farmasi, poliklinik gigi, poliklinik anak, ruang bersalin, kamar operasi, dan unit perawatan intensif (Care Unit). Fungsi dari menu ini seperti di rumah sakit lain adalah untuk membantu calon pasien atau keluarga menilai apakah RS Mentari memenuhi kebutuhan kesehatan mereka sebelum datang: apakah tersedia unit perawatan yang dibutuhkan, layanan spesialis, atau fasilitas pendukung seperti farmasi dan laboratorium. Dengan menampilkan fasilitas secara jelas, menu ini memudahkan user dalam membuat keputusan berdasarkan kebutuhan medis mereka.

Pada Gambar 3.11 RS Mentari menawarkan layanan radiologi yang memungkinkan pemeriksaan penunjang berupa pemindaian/rontgen/jenis radiologi lain sehingga pasien dengan masalah diagnostik dapat diperiksa secara tepat. Kemudian tersedia laboratorium untuk pemeriksaan lab seperti, darah, urine, atau pemeriksaan lainnya yang mendukung diagnosa atau tindak lanjut medis. Untuk kemudahan pengobatan dan peresapan obat, RS Mentari menyediakan fasilitas farmasi, memudahkan pasien mendapatkan obat yang diresepkan tanpa perlu pergi ke luar rumah sakit. Ada juga poliklinik gigi

bagi mereka yang butuh layanan kesehatan gigi dan mulut. Bagi pasien anak, tersedia poliklinik anak penting untuk perawatan kesehatan anak/pediatri. RS Mentari memiliki ruang bersalin, yang berarti melayani ibu hamil dan persalinan cocok untuk pasien yang membutuhkan layanan kebidanan. Selain itu tersedia ruang operasi, bagi pasien yang memerlukan tindakan bedah atau operasi medis. Untuk pasien yang butuh penanganan intensif atau perawatan di kamar khusus, ada "Care Unit" yang menunjukkan bahwa fasilitas perawatan intensif tersedia di rumah sakit ini.



Gambar 3.11 Tampilan Menu Fasilitas

Layanan gawat darurat RS Mentari beroperasi 24 jam setiap hari, artinya layanan gawat darurat siap beroperasi kapan saja tanpa henti untuk melayani pasien dalam kondisi mendesak. Dengan dukungan ketersediaan waktu ini, pasien dan keluarga mendapatkan kepastian bahwa pertolongan tetap dapat diakses di luar jam kerja biasa ketika situasi darurat terjadi. Di

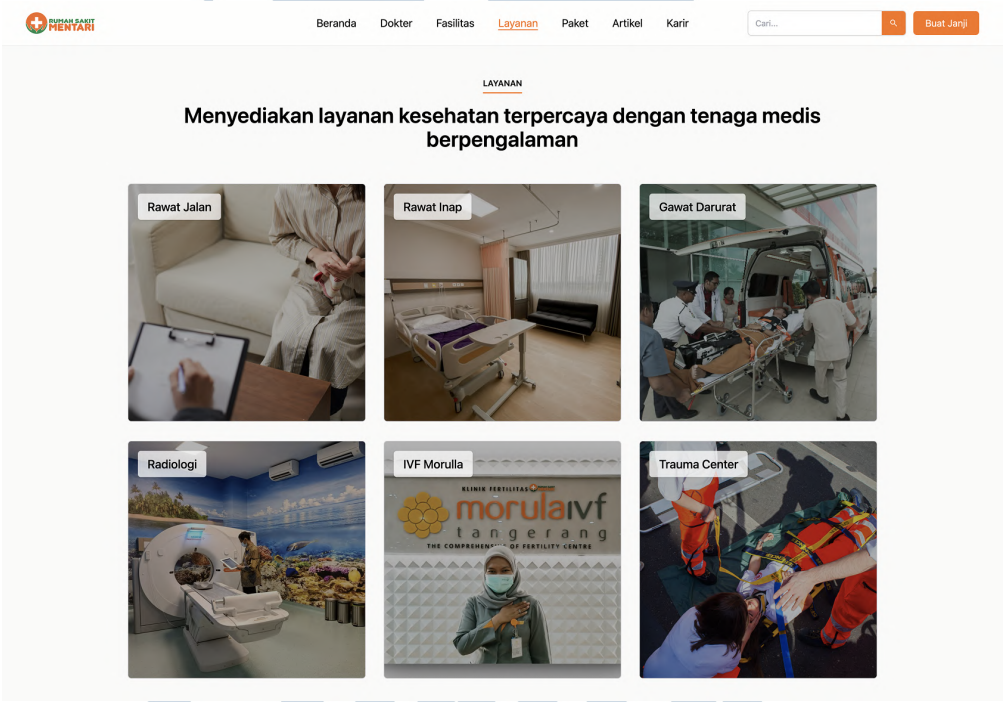
samping itu, RS Mentari menyediakan berbagai fasilitas medis pendukung seperti layanan radiologi, laboratorium, farmasi, poliklinik, ruang bersalin, kamar operasi, dan unit perawatan intensif yang membentuk rangkaian layanan komprehensif untuk menjawab beragam kebutuhan medis. Setiap fasilitas dirancang agar saling terintegrasi sehingga proses penanganan, mulai dari pemeriksaan awal hingga tindakan lanjutan, dapat berjalan lebih cepat tanpa memerlukan rujukan ke tempat lain. Seluruh layanan tersebut diorganisasikan secara terstruktur dan disajikan secara jelas, ringkas, serta mudah dipahami melalui menu fasilitas pada *website* rumah sakit sehingga memudahkan pengguna dalam mengenali jenis layanan yang tersedia.

4. Tampilan Menu Layanan

Halaman Layanan pada *website* Mentari Hospital berfungsi sebagai media utama untuk menyampaikan informasi mengenai jenis-jenis pelayanan kesehatan yang tersedia bagi pasien dan masyarakat umum. Halaman ini dirancang untuk memberikan gambaran menyeluruh mengenai cakupan layanan medis yang dimiliki rumah sakit, mulai dari pelayanan dasar hingga layanan spesialis dan penanganan kondisi darurat. Melalui tampilan visual dan struktur informasi yang sistematis, pengguna *website* dapat dengan mudah memahami fungsi masing-masing layanan sebelum memutuskan untuk datang langsung ke rumah sakit. Tampilan halaman Layanan pada *website* Mentari Hospital dapat dilihat pada Gambar 3.12.

Berdasarkan Gambar 3.12, halaman Layanan pada *website* Mentari Hospital dirancang sebagai halaman informatif yang menampilkan keseluruhan jenis pelayanan medis utama yang tersedia di rumah sakit. Halaman ini berperan penting sebagai media komunikasi antara institusi rumah sakit dan pengguna *website*, khususnya pasien atau keluarga pasien yang membutuhkan informasi awal sebelum mengakses layanan secara langsung. Penyajian layanan dilakukan secara visual melalui pembagian section yang jelas, sehingga pengguna dapat dengan mudah mengenali setiap jenis layanan yang ditawarkan. Desain halaman ini juga bertujuan untuk

meningkatkan kejelasan informasi dan mengurangi kebingungan pengguna dalam memahami cakupan pelayanan rumah sakit. Selain itu, halaman Layanan menjadi representasi digital dari kemampuan operasional rumah sakit dalam menyediakan layanan medis dasar hingga layanan spesialis. Halaman ini tidak hanya bersifat informatif, tetapi juga mendukung citra profesional rumah sakit di ranah digital, sebagaimana ditampilkan pada Gambar 3.12.



Gambar 3.12 Tampilan Menu Fasilitas

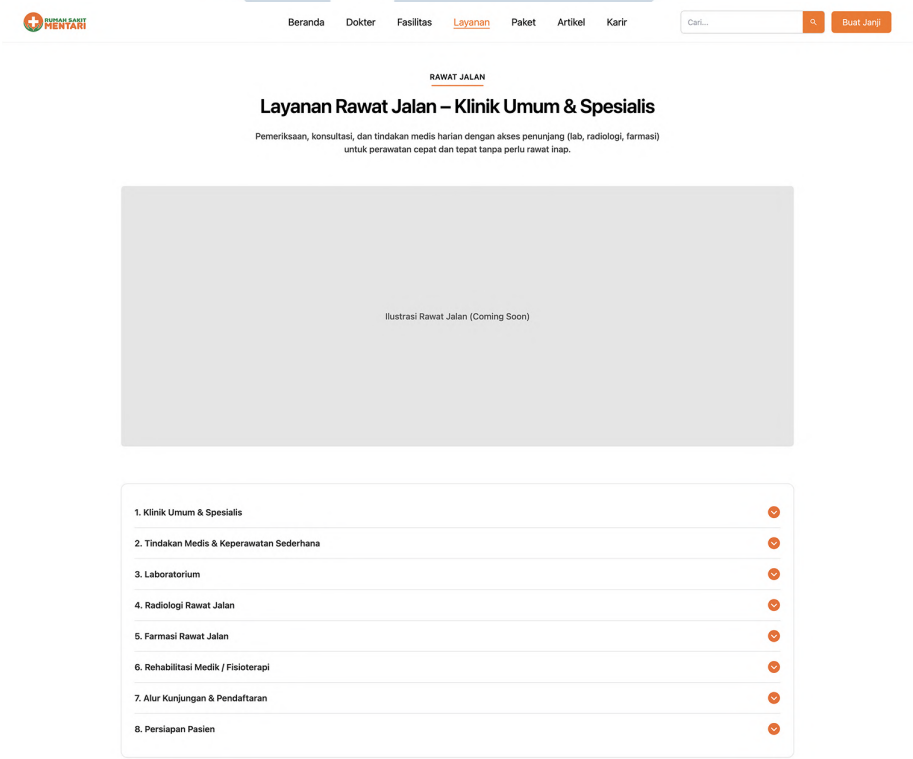
Layanan Rawat Jalan pada *website* Mentari merupakan fasilitas pelayanan medis yang ditujukan bagi pasien yang membutuhkan pemeriksaan kesehatan, konsultasi dokter, serta tindakan medis tanpa harus menjalani perawatan inap. Layanan ini memungkinkan pasien untuk mendapatkan penanganan medis dan kembali pulang pada hari yang sama. Melalui rawat jalan, pasien dengan kondisi kesehatan ringan hingga sedang dapat memperoleh layanan medis secara efektif dan efisien. Website Mentari menampilkan layanan ini sebagai solusi pelayanan kesehatan yang praktis dengan tetap mengutamakan kualitas penanganan medis. Dengan sistem pelayanan yang terorganisir, rawat jalan menjadi salah satu layanan utama

yang banyak digunakan oleh pasien. Pada halaman Rawat Jalan, *website* Mentari menyediakan berbagai pilihan layanan klinik, mulai dari klinik umum hingga klinik spesialis. Pasien dapat berkonsultasi dengan dokter umum maupun dokter spesialis seperti penyakit dalam, anak, bedah, kandungan, saraf, jantung, THT, mata, kulit dan kelamin, serta gigi. Keberagaman layanan ini menunjukkan bahwa rawat jalan di Mentari dirancang untuk memenuhi kebutuhan medis pasien secara menyeluruh. Informasi layanan yang disajikan pada *website* membantu pasien memahami jenis pelayanan yang tersedia sebelum melakukan kunjungan. Hal ini juga mencerminkan pemanfaatan *website* sebagai media informasi kesehatan yang mudah diakses oleh masyarakat.

Selain konsultasi dokter, layanan rawat jalan di Mentari juga dilengkapi dengan tindakan medis dan layanan penunjang. Tindakan keperawatan sederhana seperti perawatan luka, injeksi, dan nebulisasi dapat dilakukan langsung dalam layanan rawat jalan. Pasien juga dapat menjalani pemeriksaan penunjang seperti laboratorium dan radiologi untuk membantu proses diagnosis. Setelah pemeriksaan selesai, pasien akan mendapatkan resep obat yang dapat langsung ditebus melalui layanan farmasi yang terintegrasi. Dengan alur pelayanan yang terstruktur dari awal hingga akhir, layanan Rawat Jalan di Mentari bertujuan memberikan kemudahan, kenyamanan, serta efisiensi waktu bagi pasien.

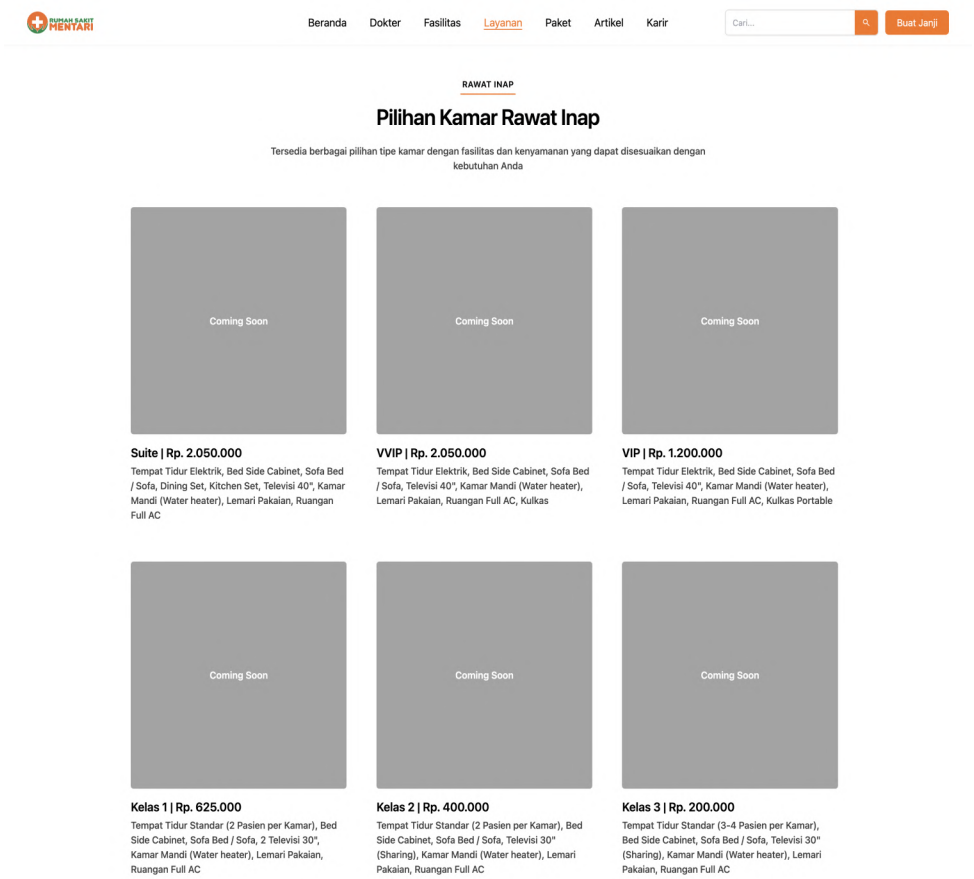
Layanan Rawat Inap pada *website* Mentari ditujukan bagi pasien yang memerlukan perawatan medis secara berkelanjutan dengan pengawasan tenaga kesehatan selama 24 jam. Pada layanan ini, pasien akan menjalani masa perawatan dengan menginap di fasilitas rumah sakit sesuai dengan kondisi medis dan rekomendasi dokter. Halaman Rawat Inap pada *website* Mentari secara khusus menampilkan informasi mengenai pilihan kamar yang tersedia sebagai bagian dari layanan perawatan pasien. Informasi tersebut disajikan untuk memberikan gambaran awal kepada pasien dan keluarga mengenai fasilitas yang akan diperoleh selama menjalani rawat inap. Dengan

penyajian ini, *website* berfungsi sebagai media informasi yang transparan terkait layanan perawatan inap yang disediakan. *Website* Mentari menampilkan beberapa kategori kamar rawat inap, mulai dari *Suite*, VVIP, VIP, Kelas 1, Kelas 2, hingga Kelas 3. Setiap kategori kamar memiliki fasilitas yang berbeda sesuai dengan tingkat kenyamanan dan privasi yang ditawarkan. Kamar dengan kategori *Suite* dan VVIP dilengkapi dengan fasilitas yang lebih lengkap, seperti tempat tidur elektrik, sofa bed, televisi, lemari pakaian, kamar mandi dengan *water heater,* serta ruang yang lebih luas untuk pasien dan pendamping. Sementara itu, kamar kelas 1, 2, dan 3 menyediakan fasilitas standar seperti tempat tidur pasien, AC, televisi, dan kamar mandi, dengan kapasitas penghuni yang lebih dari satu pasien dalam satu ruangan. Perbedaan fasilitas ini memberikan fleksibilitas bagi pasien dan keluarga dalam memilih kamar sesuai dengan kebutuhan dan kemampuan finansial.



Gambar 3. 13 Tampilan Layanan Rawat Jalan

Selain fasilitas kamar, layanan rawat inap di Mentari juga mencakup pemantauan kondisi pasien secara berkala oleh tenaga medis serta dukungan layanan penunjang selama masa perawatan. Pasien rawat inap mendapatkan pelayanan keperawatan, pemberian obat sesuai resep dokter, serta akses terhadap pemeriksaan penunjang medis apabila diperlukan. Seluruh layanan tersebut dirancang untuk mendukung proses penyembuhan pasien secara optimal selama menjalani perawatan di rumah sakit. Dengan sistem layanan yang terstruktur dan pilihan kamar yang beragam, layanan Rawat Inap pada *website* Mentari mencerminkan upaya rumah sakit dalam memberikan pelayanan kesehatan yang aman, nyaman, dan sesuai dengan standar pelayanan medis.



Gambar 3. 14 Tampilan Layanan Rawat Inap

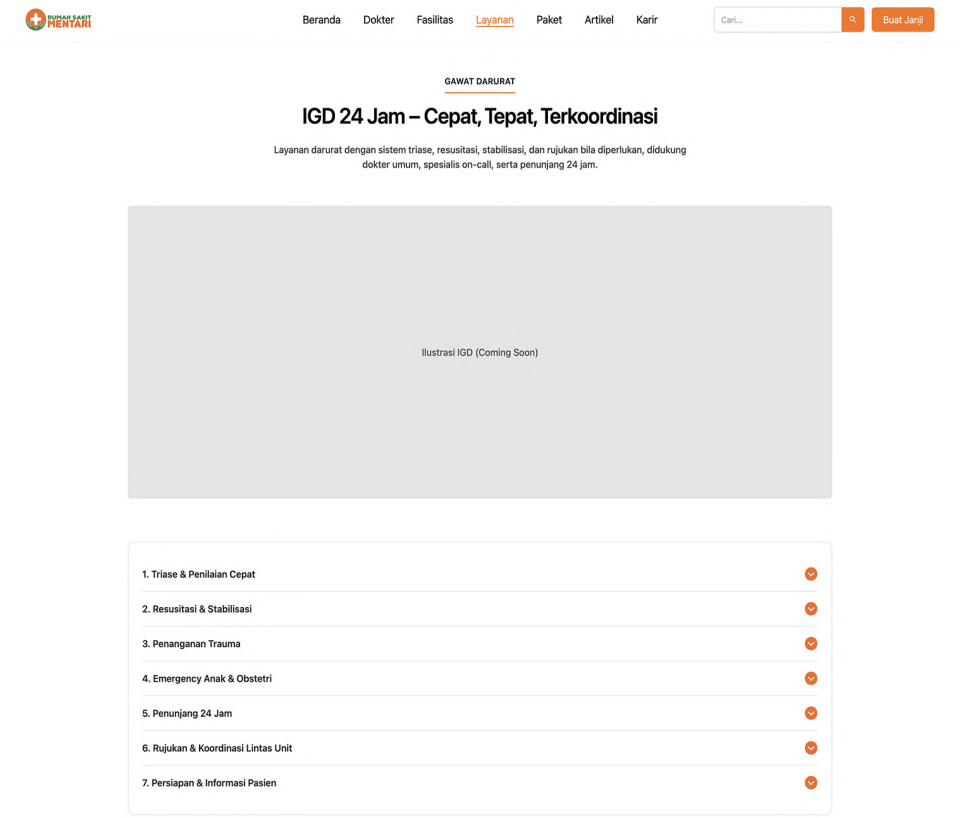
Pada Gambar 3.15, layanan Instalasi Gawat Darurat (IGD) pada website Rumah Sakit Mentari ditunjukkan sebagai unit pelayanan yang beroperasi 24

jam untuk menangani kondisi medis darurat yang membutuhkan tindakan cepat. Layanan ini berperan sebagai pintu masuk utama bagi pasien dengan kondisi kritis, baik akibat penyakit akut, kecelakaan, maupun keadaan medis lain yang berpotensi mengancam keselamatan jiwa. Pada tahap awal pelayanan, pasien akan melalui proses *triase* untuk menentukan tingkat kegawatan berdasarkan kondisi klinis yang dialami. Proses *triase* ini bertujuan untuk memastikan pasien dengan kondisi paling serius mendapatkan prioritas penanganan. Dengan sistem tersebut, IGD Mentari berupaya memberikan respon medis yang cepat, tepat, dan terstruktur.

Dalam pelaksanaannya, layanan gawat darurat di Mentari mencakup berbagai tindakan medis emergensi, mulai dari resusitasi, stabilisasi kondisi pasien, hingga penanganan trauma dan kegawatdaruratan medis lainnya. Pasien dengan kondisi seperti gangguan pernapasan, penurunan kesadaran, cedera berat, atau perdarahan akan mendapatkan penanganan sesuai dengan standar prosedur operasional yang berlaku. Pelayanan di unit IGD didukung oleh tenaga medis profesional, termasuk dokter dan perawat, yang siaga selama 24 jam. Selain itu, IGD Mentari juga didukung oleh dokter spesialis yang bersifat on-call untuk menangani kasus-kasus tertentu sesuai kebutuhan klinis pasien. Dukungan fasilitas penunjang seperti laboratorium, radiologi, farmasi, serta akses ke layanan bank darah memungkinkan proses diagnosis dan tindakan medis dilakukan secara cepat dan akurat.

Layanan gawat darurat di Mentari juga berfungsi sebagai penghubung ke layanan lanjutan apabila kondisi pasien memerlukan perawatan lebih intensif. Pasien yang membutuhkan perawatan lanjutan dapat dirujuk ke ruang rawat inap, ruang perawatan intensif (ICU), atau kamar operasi sesuai dengan hasil evaluasi medis. Koordinasi antar unit dilakukan secara terintegrasi untuk memastikan kontinuitas pelayanan dan keselamatan pasien tetap terjaga. Website Mentari menampilkan layanan gawat darurat sebagai bagian penting dari sistem pelayanan rumah sakit yang berfokus pada kecepatan, kesiapsiagaan, dan ketepatan penanganan medis. Dengan demikian, layanan IGD Mentari mencerminkan komitmen rumah sakit dalam

menyediakan pelayanan gawat darurat yang profesional, responsif, dan berorientasi pada keselamatan pasien.



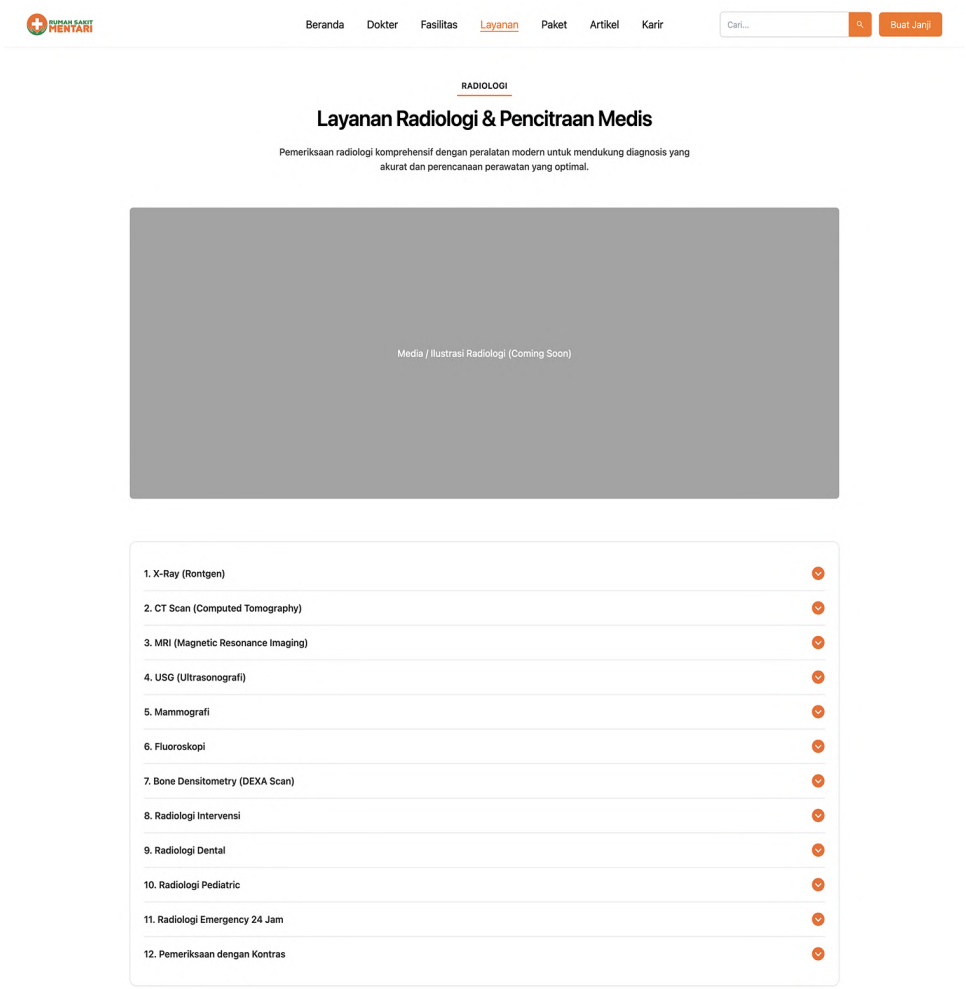
Gambar 3.15 Tampilan Layanan Gawat Darurat

Layanan Radiologi & Pencitraan Medis yang disediakan oleh Mentari merupakan bagian dari fasilitas penunjang medis yang berfokus pada pemeriksaan pencitraan tubuh untuk mendukung diagnosis dan perencanaan perawatan pasien secara akurat. Pada Gambar 3.14 halaman layanan ini, Radiologi ditampilkan sebagai unit pelayanan yang menggunakan peralatan modern dan teknologi pencitraan canggih untuk membantu dokter dalam mengevaluasi kondisi anatomis dan fisiologis pasien. Pemeriksaan radiologi memiliki peran penting dalam proses penegakan diagnosis karena hasil pencitraan memberikan gambaran visual organ dan struktur tubuh yang tidak dapat ditangkap secara langsung melalui pemeriksaan fisik saja. Website Mentari menyajikan layanan ini secara komprehensif agar

pasien, keluarga, dan tenaga medis memahami jenis pemeriksaan yang tersedia serta manfaat klinis masing-masing teknologi pencitraan.

Jenis pemeriksaan yang ditampilkan mencakup X-Ray (Rontgen), yang merupakan pemeriksaan pencitraan dasar dengan sinar-X untuk melihat kondisi tulang, sendi, dan organ dalam; CT Scan (*Computed Tomography*), yang menghasilkan gambar penampang tubuh 3 dimensi untuk deteksi kelainan internal; serta MRI (*Magnetic Resonance Imaging*) yang memanfaatkan medan magnet dan gelombang radio untuk menghasilkan citra detail jaringan lunak, tulang, dan organ tubuh lainnya. Selain itu, layanan Radiologi Mentari juga mencakup USG (Ultrasonografi), sebuah pemeriksaan tanpa radiasi yang menggunakan gelombang suara frekuensi tinggi untuk melihat organ tubuh secara real-time, serta Mammografi sebagai pemeriksaan khusus payudara untuk deteksi dini kelainan. Beberapa pemeriksaan lanjutan seperti Fluoroskopi untuk visualisasi gerakan organ secara real-time dan Bone Densitometry (DEXA Scan) untuk mengukur kepadatan mineral tulang turut ditampilkan sebagai bagian dari spektrum layanan yang tersedia.

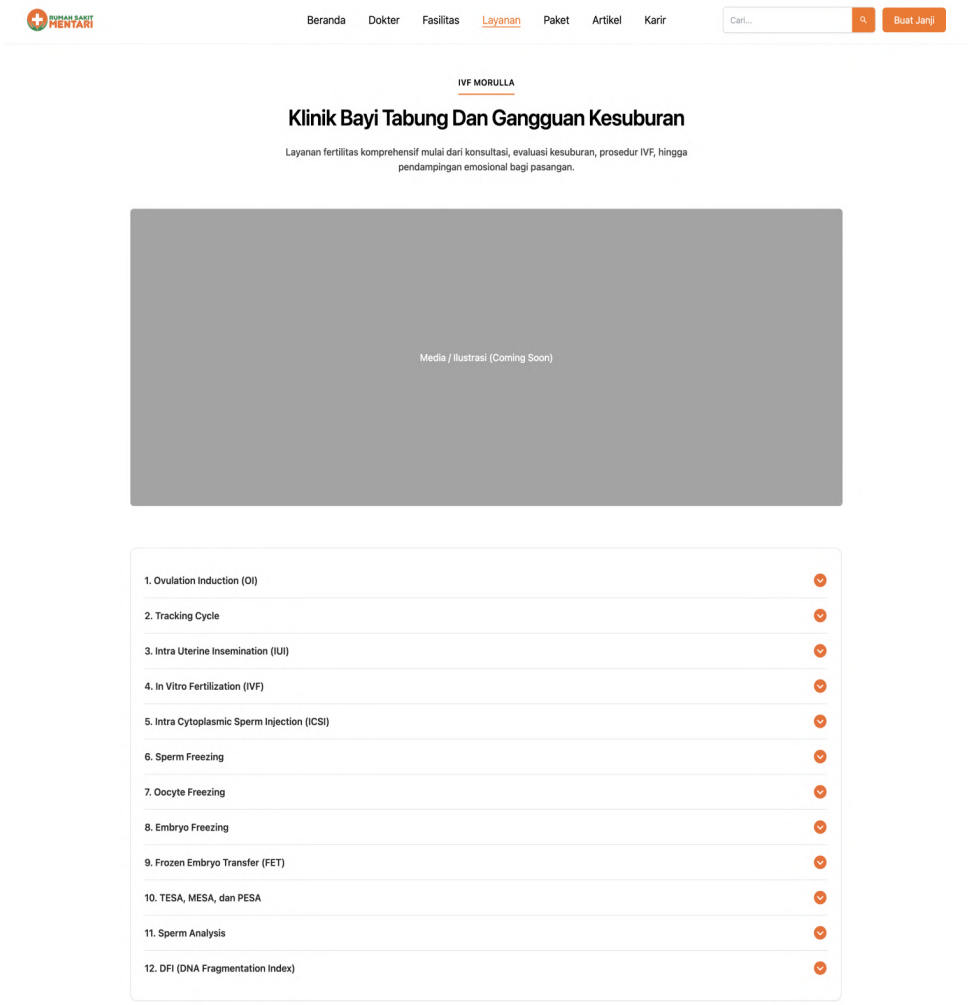
Di samping pemeriksaan diagnostik konvensional dan lanjutan, layanan Radiologi & Pencitraan Medis Mentari juga mencakup teknik pemeriksaan khusus seperti Radiologi Intervensi, yaitu prosedur minimal invasif yang dibantu panduan pencitraan untuk diagnosis dan pengobatan tertentu, serta Radiologi Dental yang mendukung diagnosa dan perencanaan perawatan pada kasus gigi dan rahang. Layanan ini dirancang untuk mendukung kebutuhan pemeriksaan pada pasien dewasa maupun anak, termasuk layanan radiologi darurat yang tersedia 24 jam untuk kasus-kasus kritis. Informasi mengenai jenis pemeriksaan dan persiapan yang diperlukan untuk beberapa pemeriksaan tertentu turut disajikan untuk meningkatkan pemahaman pasien serta memaksimalkan hasil diagnosis yang akurat. Dengan tersedianya berbagai modalitas pencitraan yang terintegrasi, layanan Radiologi Mentari mendukung pendekatan medis yang informatif, terukur, dan berbasis bukti guna mendukung keputusan klinis serta keseluruhan proses perawatan pasien.



Gambar 3.16 Tampilan Layanan Radiologi

Pada Gambar 3.17 halaman layanan IVF Morulla mencerminkan ketersediaan layanan kesehatan reproduksi berbantu yang bersifat spesialis. Layanan ini ditujukan bagi pasangan yang mengalami kesulitan dalam memperoleh keturunan secara alami dan memerlukan bantuan teknologi medis. Proses IVF melibatkan serangkaian tahapan medis yang kompleks, mulai dari stimulasi hormon, pemantauan perkembangan sel telur, pengambilan sel telur, hingga proses pembuahan di laboratorium. Embrio yang terbentuk kemudian dipantau hingga mencapai tahap perkembangan tertentu sebelum dipindahkan kembali ke rahim. Layanan ini memerlukan fasilitas khusus serta tenaga medis yang memiliki kompetensi di bidang fertilisasi berbantu. Dengan menampilkan layanan IVF Morulla pada

halaman Layanan, *website* ini menunjukkan bahwa Mentari Hospital memiliki kapabilitas untuk menyediakan layanan medis tingkat lanjut yang tidak tersedia di semua rumah sakit.



Gambar 3.17 Tampilan Layanan IVF Morulla

Layanan Trauma Center & Rehabilitasi Komprehensif pada *website* Mentari merupakan unit pelayanan medis yang dirancang untuk menangani kasus kondisi kegawat-daruratan dan cedera kompleks dengan pendekatan multi sistem yang terintegrasi. Pada layanan ini, pasien dengan trauma akibat kecelakaan, luka berat, atau kondisi kritis lainnya ditangani melalui rangkaian proses medis yang sistematis, mulai dari respon awal di Instalasi Gawat Darurat hingga perawatan lanjutan yang melibatkan berbagai disiplin ilmu. Trauma Center beroperasi dengan pendekatan multi disiplin dan memadukan

intervensi medis, diagnostik, terapi intensif, serta rehabilitasi untuk memaksimalkan pemulihan fungsi pasien secara menyeluruh. Konsep layanan ini tidak hanya berfokus pada penanganan awal cedera tetapi juga pada pemulihan jangka panjang melalui perencanaan terapi yang komprehensif dan kolaboratif.

Tahap awal penanganan trauma di Mentari dimulai dengan layanan Instalasi Gawat Darurat (IGD) yang beroperasi 24 jam dengan tim dokter dan perawat terlatih. Proses triase dilakukan secara cepat dan tepat untuk menilai prioritas klinis berdasarkan tingkat kegawatdaruratan pasien. Tindakan awal ini didukung oleh peralatan medis penting seperti perangkat resusitasi, stretcher, dan alat penanganan cedera lain yang mendukung stabilisasi kondisi pasien. Langkah ini merupakan fase krusial dalam penanganan trauma karena efektivitas respons awal sering menentukan outcome klinis jangka pendek. IGD kemudian terintegrasi dengan unit perawatan lain untuk memfasilitasi perawatan lebih lanjut sesuai kebutuhan medis pasien.

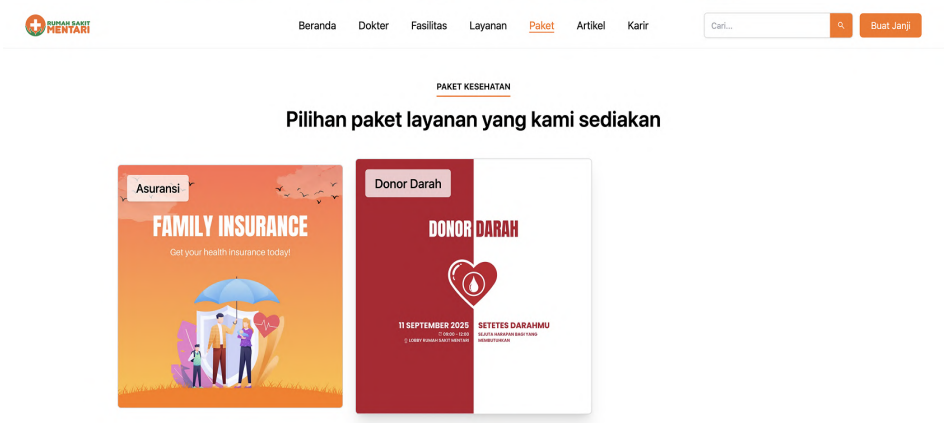
Selain layanan emergensi, Trauma Center Mentari memiliki Unit Perawatan Intensif (ICU/PICU) untuk pasien dewasa dan anak yang membutuhkan dukungan pemantauan dan perawatan intensif. Unit ini dilengkapi fasilitas medis lanjutan seperti ventilator, HFNC, serta monitor fungsi vital seperti saturasi oksigen dan tekanan darah untuk mendukung stabilisasi kondisi kritis. Seluruh proses penanganan dan perawatan intensif ini didukung oleh fasilitas penunjang yang lengkap, antara lain ruang operasi, pencitraan medis, laboratorium, serta bank darah. Peran tim medik trauma yang terdiri dari dokter spesialis berbagai bidang, termasuk bedah umum, bedah tulang, radiologi, anestesi, rehabilitasi medik, dan kedokteran darurat, memperkuat kemampuan unit ini dalam memberikan pelayanan yang komprehensif dan berbasis kebutuhan klinis pasien.

Komponen penting lain dari layanan ini adalah layanan rehabilitasi medik yang dirancang untuk mendukung pemulihan fungsi pasien pasca

trauma atau perawatan intensif. Rehabilitasi medik meliputi aspek medis spesialis, terapi fisik, edukasi, dan pendampingan untuk memaksimalkan kemampuan fungsional pasien dalam jangka menengah hingga panjang. Pendekatan integratif ini mencerminkan komitmen Trauma Center Mentari dalam memberikan pelayanan kesehatan yang holistik, di mana fokus tidak hanya pada penanganan awal tetapi juga pada proses pemulihan dan reintegrasi pasien ke aktivitas sehari-hari. Dengan demikian, layanan a ini merupakan representasi dari pelayanan kegawat daruratan yang responsif, terstruktur, dan terkoordinasi secara profesional.

5. Tampilan Menu Paket

Halaman Menu Paket pada *website* Mentari Hospital dirancang untuk menampilkan berbagai paket layanan kesehatan dalam bentuk tampilan kartu (card) yang informatif dan mudah diakses oleh pengguna. Halaman ini berfungsi sebagai sarana bagi pasien untuk melihat pilihan paket layanan yang tersedia secara ringkas sebelum memilih layanan yang sesuai dengan kebutuhannya. Setiap kartu paket ditampilkan dengan judul dan visual pendukung agar pengguna dapat dengan cepat mengenali jenis paket yang ditawarkan. Perancangan halaman ini juga bertujuan untuk meningkatkan kemudahan navigasi serta mempercepat proses pencarian informasi layanan. Tampilan halaman Menu Paket pada *website* Mentari Hospital dapat dilihat pada Gambar 3.18.



Gambar 3.18 Tampilan Paket

Berdasarkan Gambar 3.17, setiap kartu paket pada halaman Menu Paket bersifat interaktif dan dapat diklik oleh pengguna. Ketika pengguna memilih salah satu kartu paket, sistem akan menampilkan deskripsi detail mengenai paket layanan tersebut, termasuk informasi umum mengenai cakupan layanan yang diberikan. Pada halaman detail paket, tersedia tombol Appointment yang berfungsi sebagai pintu masuk bagi pasien untuk melakukan pendaftaran janji layanan. Setelah tombol Appointment dipilih, pengguna akan diarahkan ke formulir pengisian data pasien yang diperlukan untuk proses pemesanan layanan. Data yang diisikan pada formulir tersebut kemudian dikirimkan ke pihak *customer service* rumah sakit untuk ditindaklanjuti, sehingga proses penjadwalan *appointment* dapat dilakukan secara langsung dan lebih terorganisir. Dengan alur ini, sebagaimana ditunjukkan pada Gambar 3.17, halaman Menu Paket tidak hanya menyajikan informasi, tetapi juga berperan sebagai bagian dari sistem pelayanan digital yang menghubungkan pasien dengan pihak rumah sakit secara efektif.

3.2.4.2 Implementasi tampilan responsive di berbagai perangkat

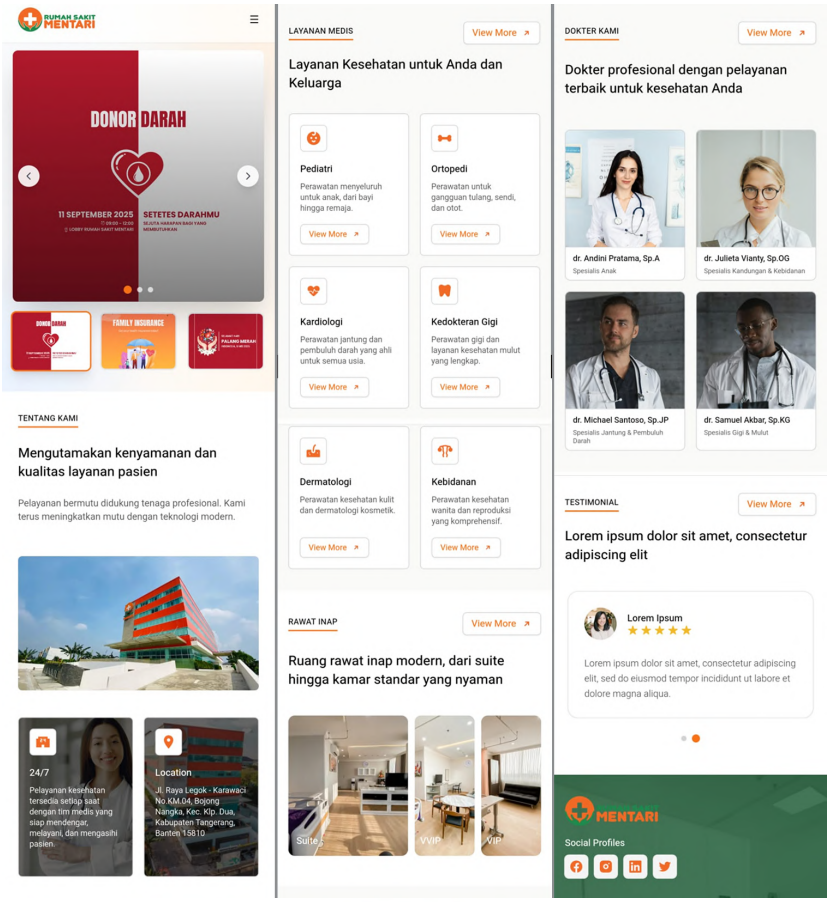
Implementasi tampilan responsif pada Gambar 3.18 *website* Mentari bertujuan untuk memastikan bahwa seluruh konten dan fitur *website* dapat diakses dengan baik oleh pengguna melalui berbagai jenis perangkat, seperti desktop, laptop, tablet, dan *smartphone*. *Website* ini dirancang menggunakan pendekatan *responsive* web *design*, di mana tata letak, ukuran elemen, serta struktur antarmuka secara otomatis menyesuaikan dengan resolusi dan ukuran layar perangkat yang digunakan. Dengan pendekatan ini, pengguna tetap mendapatkan pengalaman akses yang optimal tanpa kehilangan informasi penting, baik saat mengakses *website* melalui layar berukuran besar maupun kecil. Hal ini sangat penting mengingat layanan rumah sakit membutuhkan akses informasi yang cepat dan mudah oleh berbagai kalangan pengguna. Oleh karena itu, *responsivitas* menjadi salah satu aspek utama dalam pengembangan antarmuka *website* Mentari.

Pada tampilan desktop, *website* Mentari menampilkan struktur halaman yang lengkap dengan pemanfaatan ruang layar secara maksimal. Elemen seperti

header, menu navigasi, *banner* layanan, serta konten informasi ditampilkan secara horizontal dan terorganisir dengan baik. Menu navigasi utama ditampilkan secara penuh sehingga memudahkan pengguna dalam berpindah antar halaman layanan, seperti Rawat Jalan, Rawat Inap, Gawat Darurat, Radiologi, dan Trauma Center. Tata letak ini memberikan kemudahan bagi pengguna dalam memahami struktur informasi secara menyeluruh dalam satu tampilan layar. Penggunaan *grid* dan pembagian kolom yang konsisten menunjukkan bahwa desain *website* telah disesuaikan dengan standar antarmuka modern pada perangkat desktop.

Pada perangkat tablet dan *smartphone*, tampilan *website* Mentari mengalami penyesuaian tata letak secara dinamis. Menu navigasi yang sebelumnya ditampilkan secara horizontal akan berubah menjadi ikon hamburger menu untuk menghemat ruang layar. Konten halaman ditampilkan dalam satu kolom vertikal agar mudah dibaca dengan cara *scrolling*. Ukuran teks, tombol, dan elemen interaktif seperti kartu layanan (*card*) serta tombol aksi (*button*) disesuaikan agar tetap jelas dan mudah diakses menggunakan sentuhan jari. Penyesuaian ini bertujuan untuk meningkatkan kenyamanan pengguna *mobile*, mengingat sebagian besar pengguna mengakses informasi layanan kesehatan melalui perangkat seluler.

Selain penyesuaian tata letak, *website* Mentari juga menerapkan prinsip *responsivitas* pada elemen visual dan interaktif. Gambar, ilustrasi, serta ikon layanan akan menyesuaikan ukuran secara proporsional tanpa mengganggu komposisi halaman. Komponen seperti *card* layanan tetap mempertahankan fungsi interaktifnya, di mana pengguna dapat melakukan klik atau sentuhan untuk melihat informasi detail terkait layanan yang dipilih. Tombol-tombol penting, seperti tombol navigasi atau tindakan lanjutan, tetap terlihat jelas dan mudah dijangkau pada berbagai ukuran layar. Hal ini menunjukkan bahwa implementasi responsif tidak hanya berfokus pada tampilan visual, tetapi juga pada aspek fungsionalitas dan kegunaan.



Gambar 3.19 Tampilan Responsif Mobile

3.2.5 Membuat *backend website* rumah sakit mentari

Backend Website Rumah Sakit Mentari merupakan sebuah sistem manajemen rumah sakit berbasis web yang dirancang untuk mengelola berbagai data dan layanan secara terpusat dan terstruktur. Sistem ini dibangun menggunakan *framework* Next.js dengan *TypeScript* sebagai bahasa pemrograman utama, sehingga pengembangan aplikasi menjadi lebih rapi, terkontrol, dan minim kesalahan. *Backend* dan *frontend* diintegrasikan dalam satu *framework* yang sama melalui pendekatan API *Routes* milik Next.js, yang memungkinkan pembuatan layanan *backend* tanpa harus menggunakan server terpisah. Dengan adanya API *Routes*, setiap fitur seperti pengelolaan data dokter, layanan kesehatan, dan jadwal praktik dapat diakses melalui REST API yang berada di dalam folder *pages*/api. *Backend* dijalankan pada *port* 3001 agar terpisah dari layanan lain dan memudahkan proses pengembangan serta pengujian. Sebagai lingkungan server

lokal, sistem ini menggunakan XAMPP yang dikombinasikan dengan MySQL sebagai *database management system* untuk menyimpan dan mengelola seluruh data rumah sakit secara aman dan terorganisir.

3.2.5.1 Pembuatan *database* (implementasi MySQL/PostgreSQL)

Pembuatan dan implementasi *database* pada *Backend Website* Rumah Sakit Mentari diawali dengan pemilihan *MySQL* sebagai *Relational Database Management System* (RDBMS) yang dijalankan melalui XAMPP pada *port* standar 3306. *Database* yang digunakan diberi nama mentari_db dan dibuat menggunakan perintah SQL dengan pengaturan *character* set serta *collation* utf8mb4_unicode_ci. Pengaturan ini bertujuan untuk memastikan *database* mampu menyimpan karakter *Unicode* secara penuh, termasuk simbol khusus, *emoji*, dan karakter multibahasa tanpa mengalami kerusakan data. Seluruh struktur awal *database* didefinisikan dalam sebuah *file schema*.sql yang disimpan di folder *database* dan dapat dieksekusi melalui *phpMyAdmin* maupun *command line* MySQL. Pendekatan ini mempermudah proses *bootstrap database*, memastikan struktur tabel konsisten, dan mempercepat proses setup baik pada tahap *development* maupun pengujian.

Struktur *database* mentari_db dirancang secara relasional dengan memperhatikan keterkaitan antar data serta konsistensi informasi yang disimpan. *Database* ini terdiri dari beberapa tabel utama, yaitu *admins*, *specializations*, *doctors*, dan *doctor_schedules*, yang masing-masing memiliki peran spesifik dalam mendukung operasional sistem. Tabel *admins* digunakan untuk menyimpan data administrator dengan atribut *role* berupa *superadmin* atau admin, *permissions* dalam format JSON untuk fleksibilitas kontrol akses, serta status akun melalui *field is_active*. Tabel *specializations* menyimpan data spesialisasi medis dengan *slug* unik untuk mendukung URL yang ramah SEO, *icon* sebagai elemen visual antarmuka, serta *flag is_hidden* untuk mengatur visibilitas data. Tabel *doctors* menjadi pusat data dokter yang memiliki relasi *foreign key* ke tabel *specializations*, sementara tabel *doctor_schedules* berelasi langsung dengan tabel *doctors* menggunakan mekanisme ON DELETE CASCADE untuk menjaga konsistensi data ketika terjadi penghapusan. Seluruh tabel dilengkapi dengan *primary key auto*

increment, unique constraint untuk mencegah duplikasi data, serta *field created_at* dan *updated_at* sebagai audit trail.

Koneksi antara *backend* dan *database* diimplementasikan melalui *file database*.ts menggunakan library mysql2 dengan pendekatan *promise-based* API yang mendukung penggunaan *async/await*. Konfigurasi koneksi *database* tidak ditulis secara *hardcoded*, melainkan diambil dari *environment variables* yang disimpan pada *file .env.local,* sehingga meningkatkan keamanan dan fleksibilitas konfigurasi. Variabel seperti DB_HOST, DB_PORT, DB_USER, DB_PASSWORD, dan DB_NAME digunakan untuk menyesuaikan koneksi dengan *environment* yang digunakan. Alamat *host database* diset ke 127.0.0.1 alih-alih *localhost* untuk menghindari potensi masalah resolusi IPv6 yang sering terjadi pada konfigurasi XAMPP. *Backend* juga menerapkan mekanisme *connection pooling* dengan batas maksimum sepuluh koneksi aktif, sehingga sistem dapat menangani banyak *request* secara bersamaan tanpa membebani server atau membuat koneksi baru setiap kali permintaan masuk.

Selain koneksi dasar, sistem *database* juga dirancang dengan mekanisme *dynamic pool management* untuk meningkatkan fleksibilitas selama proses pengembangan. *Metadata* koneksi *database* disimpan pada global *state*, sehingga *backend* mampu mendeteksi perubahan konfigurasi *database* secara otomatis. Ketika terjadi perubahan kredensial *atau host database, connection pool* lama akan ditutup dan digantikan dengan *pool* baru tanpa perlu melakukan *restart* aplikasi. Pada *environment development*, koneksi *database* diverifikasi sejak aplikasi dijalankan agar kesalahan koneksi dapat langsung terdeteksi lebih awal. *Backend* juga memiliki mekanisme peningkatan pesan *error* yang mampu menampilkan informasi yang lebih informatif, seperti kesalahan *autentikasi* atau hak akses *database*, sehingga memudahkan proses *debugging* dan *troubleshooting* selama pengembangan.

Untuk menjaga keamanan dan konsistensi eksekusi *query*, *backend* menyediakan layer abstraksi khusus dalam *file database.*ts. Layer ini menyediakan

fungsi utama untuk mengambil koneksi dari *pool*, mengeksekusi *query* menggunakan *prepared statement*, serta melakukan pengecekan kesehatan koneksi *database*. Seluruh *query* dieksekusi menggunakan parameter *binding* dengan *placeholder*, sehingga *input* pengguna tidak pernah langsung digabungkan ke dalam *string* SQL. Pendekatan ini sangat penting untuk mencegah serangan SQL *injection* dan meningkatkan keamanan sistem secara keseluruhan. Setiap koneksi yang digunakan akan selalu dikembalikan ke *pool* setelah *query* selesai dijalankan, sehingga risiko kebocoran koneksi dapat dihindari. Selain itu, tersedia mode debug yang dapat diaktifkan untuk menampilkan seluruh *query* SQL ke *console*, yang sangat membantu dalam proses optimasi dan analisis performa *database*.

Database mentari_db juga dilengkapi dengan mekanisme inisialisasi data melalui proses *seeding* menggunakan *file SQL* terpisah. *File* seperti *seed_admins.sql* dan *seed_specializations*.sql digunakan untuk mengisi data awal, termasuk akun admin, daftar spesialisasi medis, serta contoh data dokter yang sudah terhubung dengan spesialisasinya. Keberadaan data *seed* ini memastikan *database* tidak dalam kondisi kosong saat pertama kali digunakan, sehingga seluruh fitur *backend* dapat langsung diuji. Data awal ini juga membantu proses demonstrasi sistem dan pengujian fungsionalitas API. Untuk penggunaan pada lingkungan produksi, sistem ini masih dapat dikembangkan lebih lanjut dengan menambahkan mekanisme *backup* rutin, migrasi skema *database*, serta pengelolaan versi struktur *database* secara otomatis.

Dari sisi keamanan dan *best practices*, implementasi *database* telah memperhatikan pemisahan konfigurasi sensitif dari *codebase* utama dengan menggunakan *environment variables. File .env.local* yang menyimpan informasi kredensial *database* tidak disertakan dalam *version control* untuk mencegah kebocoran data. Disarankan agar aplikasi tidak menggunakan akun root pada *database*, melainkan akun khusus dengan hak akses terbatas yang hanya memiliki izin pada *database mentari_db*. Pendekatan ini bertujuan untuk meminimalkan risiko apabila terjadi kompromi keamanan pada aplikasi. Selain itu, strategi *backup*

database secara berkala juga sangat disarankan untuk menjaga keberlangsungan data, terutama ketika sistem sudah digunakan pada lingkungan produksi.

3.2.5.2 Melakukan pembuatan backend (API, autentikasi, integrasi *database*)

Backend Website Rumah Sakit Mentari menggunakan Next.js API *Routes* untuk membuat *REST* API *endpoints* yang *serverless*, dimana setiap *file* di folder *pages*/api otomatis menjadi *endpoint* HTTP. Setiap API *endpoint* (seperti /api/doctors, /api/auth/login, /api/upload) dibangun dengan *handler* yang menerima *request* dan *response object*, kemudian melakukan *routing* berdasarkan HTTP *method* (GET, POST, PUT, DELETE) ke *function logic* yang sesuai. Setiap *endpoint* diintegrasikan dengan *database* layer melalui *function executeQuery*() yang menjalankan SQL *query* dengan *prepared statement* untuk mencegah SQL *injection*. *Response* API selalu dalam format JSON terstruktur berisi *success (boolean), message* (deskripsi), dan *optional* data (*payload*) atau *error field*. Semua *handler* membungkus *logic* dalam *try-catch* untuk *centralized error handling*, memastikan *error database* atau *validation* langsung ditangkap dan dikembalikan dengan HTTP status *code appropriate* (200 *success*, 400 bad *request,* 401 *unauthorized*, 500 server *error*).

Sistem *autentikasi* diimplementasikan di *login*.ts dengan validasi email/*username* dan *password* terhadap tabel *admins*, *endpoint* ini hanya *accept* POST *request* dan melakukan *query prepared statement* dengan filter *is_active* = 1 untuk memastikan hanya admin aktif yang bisa *login. Response* mengembalikan data *user* (*id, username*, email, *full_name*, role, *permissions*) tanpa password field untuk *security*. *Pada client-side, session* disimpan di *localStorage/sessionStorage*, kemudian *hook useAdminProtection*() melakukan *checking: superadmin bypass* semua halaman, sementara admin biasa harus memiliki *specific permission* untuk akses halaman tertentu. Jika *session* tidak valid atau *permission* tidak cukup, otomatis *redirect* ke /admin/*login* atau /admin/*unauthorized pattern* ini memastikan halaman admin terlindungi dari akses *unauthorized* dan hanya *accessible* oleh admin dengan *role* dan *permission* yang sesuai.

Backend menyediakan operasi CRUD lengkap untuk *resource* utama (*doctors*, *specializations*, *schedules*, layanan, dll) dengan struktur *consistent*, *GET retrieve* data (support filter by id), POST *create record* dengan validasi required *fields*, PUT *update existing record*, DELETE *remove record*. *Endpoint upload*.ts menangani *file upload* dengan *parsing* dua format *multipart form* data dan base64 data URL kemudian *sanitize filename*, simpan di folder *uploads*, dan *return* JSON berisi URL yang dapat langsung digunakan *frontend*. Setiap operasi menggunakan parameter *binding* untuk *query security, auto-manage connection pooling* dari *pool* yang *pre-configured* di *database*.ts dengan *connectionLimit*, 10 untuk optimal *performance*. *Error handling* konsisten di seluruh API, *database* error di-*enhance* dengan *hint helpful* (misalnya jika *connection denied*, *suggest create user* baru), *security practices* diterapkan seperti *exclude password* dari *response, sanitize file upload*, dan *middleware autentikasi* melindungi *endpoint* sensitif *creating robust, scalable,* dan *secure backend infrastructure* untuk aplikasi rumah sakit.

3.2.5.3 Membuat web admin panel

Admin panel *Website* Rumah Sakit Mentari dibangun dengan arsitektur modular menggunakan *React component* dan Tailwind CSS untuk *styling*. Halaman admin (*pages*/admin) terdiri dari beberapa *section* utama: /admin/login untuk *autentikasi*, /admin/index sebagai *landing* yang melakukan *permission-based routing* ke halaman sesuai akses *user*, /admin/*healthcare*, /admin/dokter, /admin/layanan, /admin/menu, /admin/testimonial, dan /admin/*adminsettings* untuk berbagai fungsi manajemen. *Layout* utama *dihandle* oleh komponen *AdminLayout* yang terdiri dari *sidebar navigation AdminSidebar* dan *content area*, dimana sidebar secara dinamis menampilkan menu *items* berdasarkan *user role* dan *permissions* (*superadmin* akses semua, admin biasa hanya lihat menu yang sesuai *permission*). Struktur ini memastikan UI *consistent* di seluruh halaman admin dan *permission-based navigation* mencegah *user* mengakses menu yang tidak berhak.

Halaman *login* (/admin/*login*) merupakan *entry point* admin panel dengan *design* yang *user-friendly* menggunakan *two-column layout*, *left* panel menampilkan *image banner*, *right* panel berisi *form login. Form* menerima *input*

email/*username* dan *password*, dengan *checkbox* "Remember Me" untuk *persistent login* (disimpan di *localStorage*), atau *temporary session* di *sessionStorage*. *Button submit* melakukan *fetch* POST ke /api/auth/login, dan jika *successful*, *response* berisi *user* data (id, username, role, permissions) yang disimpan di localStorage/sessionStorage tanpa *password field*. Jika login gagal, *error message* ditampilkan (validation error dari server), dan *form* tetap di halaman dengan *input* preserved. Setelah login sukses, *router* melakukan *push* ke /admin/*adminsettings* (untuk *superadmin*) atau halaman pertama sesuai *permission user*. *Loading state* dan *error state* dikelola untuk UX yang baik (*button* di-*disable* saat *loading*, *error message* jelas).

Dashboard admin (/admin/index) berfungsi sebagai hub yang melakukan *check session* dan *permission-based redirect* ke halaman manajemen sesuai akses *superadmin* ke *settings*, user biasa ke halaman *first accessible*. Halaman manajemen data (seperti /admin/dokter) menampilkan data dalam format tabel dengan inline CRUD actions, tombol edit membuka modal *form* untuk update data, tombol *delete* menghapus dengan confirmation, tombol *toggle* untuk hide/show *record*. Komponen *ServiceEditor* untuk layanan mengimplementasi *complex form* dengan rich text editor, *image upload* (via /api/upload endpoint), dan *multiple section management* untuk konten dinamis. Setiap halaman *include fetching* data dari API *endpoint* terkait (/api/doctors, /api/layanan/[slug], dll) menggunakan *function fetcher, state management via useState* untuk *form* data dan *loading states*, serta *error handling* dengan *try-catch*. *Form submission* melakukan PUT/POST ke API *endpoint*, dan setelah *success*, data di-*refresh* atau modal ditutup *creating smooth* data *management experience*. Admin *panel complete* dengan *TypeScript type definitions* untuk *type-safety,* menjadikan *development* lebih *robust* dan *maintainable*.

3.3 Melakukan Optimasi dan Maintenance

Seluruh aktivitas peningkatan kualitas dan keandalan situs secara berkelanjutan. Fokusnya adalah menjaga pengalaman pengguna yang konsisten,

performa sistem yang stabil, serta keamanan dan kemudahan pemeliharaan. Kegiatan inti meliputi pemantauan metrik kinerja (*Core Web Vitals*, TTFB, LCP, FID), *profiling* API untuk mendeteksi *bottleneck*, penataan ulang arsitektur informasi agar konten mudah ditemukan, serta pembaruan rutin dependensi untuk kompatibilitas dan penutupan celah keamanan. Di sisi data, dilakukan audit indeks, normalisasi skema seperlunya, serta pengaturan *connection pooling* agar *query* tetap efisien saat beban meningkat. *Observabilitas* dipenuhi dengan *logging* terstruktur, *health checks*, dan *error monitoring* sehingga tim dapat melakukan *troubleshooting* cepat, menyiapkan *rollback* bila perlu, dan menghindari *downtime* berkepanjangan. *Maintenance* juga mencakup siklus *backup–restore* teruji, dokumentasi konfigurasi, dan siklus rilis terukur (*changelogs*, *post-release checks*) agar setiap perubahan bisa dilacak dan dikendalikan.

3.3.1.1 Melakukan optimasi navigasi, herarki dan performa website (implementasi CDN)

Navigasi utama pada *website* dirancang melalui komponen *navbar* yang memuat menu “Beranda”, “Dokter”, “Fasilitas”, “Layanan”, “Paket”, serta *placeholder* “Artikel” dan “Karir”. Struktur navigasi ini ditampilkan secara konsisten baik pada perangkat desktop maupun *mobile*, sehingga pengguna tidak perlu beradaptasi ulang saat berpindah perangkat. Setiap menu memiliki *state* aktif yang ditandai dengan garis bawah (*underline*) dan warna aksen, berfungsi sebagai umpan balik visual agar pengguna mengetahui posisi halaman yang sedang diakses. Pada tampilan desktop, menu disusun di bagian tengah *navbar*, sedangkan pada perangkat *mobile* navigasi diubah menjadi *drawer* menu yang muncul melalui tombol *toggle*, sehingga tetap mudah digunakan pada layar yang lebih sempit. Selain menu utama, terdapat tombol *Call to Action* (CTA) “Buat Janji” yang selalu ditampilkan secara konsisten. Keberadaan tombol ini menegaskan alur utama yang ingin dicapai pengguna, yaitu melakukan pendaftaran atau janji layanan kesehatan. Fitur pencarian juga terintegrasi langsung di dalam *navbar* pada versi desktop, dan ditempatkan di dalam *drawer* pada versi *mobile*. Mekanisme pencarian memanfaatkan data dari *searchIndex*, sehingga pengguna dapat langsung menuju

konten yang relevan tanpa harus menelusuri menu satu per satu. Walaupun fitur *breadcrumb* belum digunakan, struktur hierarki halaman tetap terjaga melalui konsistensi penamaan URL, seperti /menu/dokter dan /menu/layanan, serta adanya pengaturan *redirect* di *next.config.mjs* untuk menyeragamkan penulisan URL, khususnya pada halaman jadwal dokter.

Pada bagian atas halaman, hierarki konten ditampilkan melalui komponen *PageCover* yang berfungsi sebagai area hero. Di bagian ini, judul utama dan deskripsi singkat ditampilkan terlebih dahulu sebagai informasi inti, kemudian diikuti oleh komponen pencarian layanan. Susunan ini membantu pengguna memahami konteks halaman sebelum melakukan interaksi lebih lanjut. Judul ditata dalam dua baris dengan penekanan warna aksen pada kata tertentu untuk meningkatkan keterbacaan dan daya tarik visual. Sementara itu, deskripsi yang lebih panjang disembunyikan pada tampilan *mobile* agar halaman tetap ringkas dan nyaman dibaca. Elemen *slider* pada hero menampilkan gambar layanan dan fasilitas sebagai visual pendukung, yang datanya dapat diambil secara dinamis melalui API /api/*header*, sehingga konten dapat diperbarui tanpa perlu perubahan langsung pada kode program. Penggunaan *heading* dan elemen semantik membantu membangun alur baca yang jelas, dimulai dari *headline*, dilanjutkan deskripsi, dan kemudian elemen interaktif seperti pencarian atau CTA. Konsistensi penggunaan slug dan penamaan *path* pada halaman layanan dan dokter juga mendukung navigasi yang lebih mudah diprediksi oleh pengguna, sehingga pengalaman menjelajah situs menjadi lebih intuitif.

Performa *front-end* dijaga melalui beberapa pendekatan yang efisien. Komponen seperti *navbar* dan *page cover* menggunakan pengelolaan *state* yang minimal untuk mengurangi beban *render*. Gambar-gambar hero dan aset statis diambil dari folder publik atau konfigurasi API, sehingga tidak memerlukan proses pemanggilan data yang berat. Fitur pencarian memanfaatkan indeks data lokal, memungkinkan respons yang cepat tanpa harus mengakses server setiap kali pengguna mengetik. Selain itu, Next.js secara bawaan menerapkan *code-splitting*, sehingga hanya kode yang diperlukan pada halaman tertentu saja yang dimuat.

Pengaturan *responsivitas* dilakukan dengan *utility class* dari *Tailwind* CSS, yang membuat pengelolaan tampilan lebih ringan dan terstruktur. Meskipun saat ini belum menggunakan CDN khusus, performa masih bergantung pada mekanisme *caching* bawaan Next.js untuk aset statis dan efisiensi pemanggilan API, dengan peluang pengembangan lebih lanjut melalui distribusi ke *edge* atau CDN di masa mendatang.

3.3.1.2 Implementasi content, melakukan debugging, troubleshooting, dam optimasi *database*

Pengelolaan konten dinamis pada sistem *website* ini dilakukan melalui API yang membaca dan menulis data langsung ke dalam berkas, sehingga perubahan konten dapat dilakukan tanpa perlu melakukan *redeploy* aplikasi. Salah satu implementasinya terdapat pada header.ts yang mengelola file header.json untuk menyimpan judul, deskripsi, daftar *slide*, serta pengaturan *autoplay* pada bagian *header* halaman. *Endpoint* ini menyediakan metode GET untuk membaca data, POST untuk menambah atau memperbarui *slide*, serta PUT untuk mengganti seluruh konfigurasi *header.* Apabila *file* data belum tersedia atau mengalami kerusakan, sistem secara otomatis membuat *file* baru dengan nilai *default* agar konten tetap dapat ditampilkan. Pendekatan berbasis *file* ini dipilih karena sederhana, mudah dikelola, dan cukup efektif untuk konten yang tidak terlalu kompleks. Dengan mekanisme tersebut, tim pengelola dapat memperbarui tampilan dan informasi tanpa bergantung pada migrasi *database*.

Pendekatan file-based yang sama juga diterapkan pada pengelolaan jadwal dokter melalui doctor-schedules.ts yang menyimpan data dalam file doctor_schedules.json. API ini mendukung operasi GET, POST, PUT, dan DELETE sehingga jadwal dapat ditambah, diubah, atau dihapus sesuai kebutuhan. Sistem menyediakan fitur pembuatan *slot* jadwal otomatis dengan interval 30 menit, termasuk pengaturan jeda istirahat dan pengecualian pada tanggal tertentu. Dengan adanya fitur ini, pengaturan jadwal menjadi lebih konsisten dan mengurangi kesalahan *input* manual. Seluruh perubahan jadwal dapat langsung diakses oleh

sistem tanpa perlu proses *build* ulang aplikasi. Hal ini memberikan fleksibilitas tinggi dalam pengelolaan layanan dokter sehari-hari.

Dari sisi stabilitas dan performa, sistem dilengkapi dengan mekanisme *debugging*, pengecekan koneksi *database*, serta optimasi eksekusi *query*. Setiap *handler* API dibungkus dengan *try-catch* dan mengembalikan pesan *error* yang jelas ketika terjadi kegagalan, sehingga memudahkan proses penelusuran masalah. Tersedia *endpoint* test-db.ts untuk memastikan koneksi *database* berjalan dengan baik dan membantu diagnosis masalah kredensial atau akses. Koneksi *database* dikelola menggunakan *connection pool* dan *prepared statements* untuk menjaga efisiensi serta keamanan *query*. Struktur skema *database* juga dirancang dengan *constraint* seperti UNIQUE dan *foreign key* untuk menjaga konsistensi data. Kombinasi antara pengelolaan *file*, pengamanan API, dan optimasi *database* ini memberikan fondasi sistem yang stabil, mudah dipelihara, dan siap digunakan dalam operasional sehari-hari..

3.3.1.3 Melakukan implementasi keamanan serta testing dan deployment ke server

Sebagai bagian penting dari pengembangan sistem, aspek keamanan dirancang untuk memastikan hanya pengguna yang berwenang yang dapat mengakses fitur administratif. Mekanisme autentikasi admin disediakan melalui *endpoint login* yang memverifikasi email atau *username* serta *password* pada tabel *admins*. Sistem hanya mengizinkan akun dengan status aktif dan mengembalikan data profil tanpa menyertakan *field password* demi menjaga kerahasiaan informasi. Pengaturan hak akses diterapkan menggunakan konsep *role* (*superadmin* dan admin) serta *permissions* yang disimpan dalam format JSON. Di sisi klien, hak akses ini diperiksa melalui *hook useAdminProtection* untuk membatasi akses halaman berdasarkan kewenangan pengguna. *Superadmin* memiliki akses penuh ke seluruh fitur, sedangkan admin biasa hanya dapat mengakses fitur sesuai *permission* yang dimilikinya.

Untuk menjaga keamanan data dan stabilitas sistem, akses ke *database* dan proses unggah *file* juga dikontrol dengan ketat. Seluruh interaksi *database* menggunakan *prepared statements* melalui fungsi *executeQuery*, yang efektif mencegah serangan SQL *injection*. Proses unggah *file* ditangani oleh *endpoint upload* dengan penerapan sanitasi nama *file*, pembatasan ukuran, serta penyimpanan pada direktori publik yang telah ditentukan. Parser bawaan Next.js dinonaktifkan agar *formidable* dapat memproses data *multipart* secara lebih aman. Selain itu, *endpoint ensureAdminsTable* digunakan untuk memastikan tabel *admins* tersedia dan melakukan *seeding* akun *superadmin* serta admin *default*. Proses ini dilindungi agar hanya dijalankan satu kali dalam satu proses aplikasi, sehingga mencegah konflik atau eksekusi berulang.

Dari sisi pengujian dan penanganan masalah, sistem dilengkapi dengan berbagai mekanisme untuk memudahkan proses *debugging*. *Endpoint* test-db disediakan untuk memverifikasi koneksi *database* dan mengecek kredensial melalui perintah SHOW DATABASES. Seluruh API dibungkus dengan try-catch dan mengembalikan status HTTP beserta pesan error yang jelas apabila terjadi kegagalan. Kesalahan yang muncul juga dicatat ke dalam log server untuk membantu penelusuran masalah. *Guard* global seperti MENTARI_ADMINS_ENSURED mencegah operasi skema *database* dijalankan berulang kali saat *runtime*. Pada tahap *deployment*, aplikasi berjalan sebagai Next.js app dengan API *routes* internal, aset statis dari folder *public*, serta pengaturan *redirect* di next.*config*.mjs, sementara tingkat keamanan akhir sangat bergantung pada konfigurasi *environment*, penggunaan HTTPS, dan pengaturan *hosting* yang diterapkan.

3.4 Kendala yang Ditemukan

Selama pelaksanaan kerja magang pada pengembangan *website* Rumah Sakit Mentari, proses pengerjaan tidak terlepas dari berbagai kendala yang muncul baik pada tahap perencanaan, pengembangan, maupun implementasi sistem. Kendala-kendala ini muncul seiring dengan upaya penyesuaian antara kebutuhan pengguna, kondisi sistem sebelumnya, serta keterbatasan sumber daya yang tersedia.

Identifikasi kendala dilakukan berdasarkan pengalaman langsung selama pelaksanaan kerja magang serta hasil diskusi dan evaluasi dengan pihak terkait. Adanya kendala tersebut menjadi faktor yang perlu diperhatikan agar tahapan pengembangan *website* dapat berjalan dengan lebih terarah dan sesuai dengan tujuan yang telah ditetapkan.

1. Keterbatasan dokumentasi sistem *website* sebelumnya

   Website lama Rumah Sakit Mentari dikembangkan tanpa dokumentasi teknis yang lengkap dan terstruktur. Tidak tersedia penjelasan tertulis mengenai arsitektur sistem, alur data, maupun konfigurasi fitur yang digunakan. Kondisi ini menyebabkan mahasiswa harus melakukan analisis mandiri terhadap sistem yang sudah berjalan sebelum memulai proses pengembangan. Tahapan analisis membutuhkan durasi yang cukup panjang karena dilaksanakan melalui pengamatan langsung dan percobaan teknis. Selain itu, keterbatasan dokumentasi juga meningkatkan risiko terjadinya kesalahan pemahaman terhadap fungsi sistem. Hal ini berdampak pada lamanya tahap awal pengembangan *website*.

2. Kendala dalam pengumpulan dan perubahan kebutuhan sistem

   Proses pengumpulan kebutuhan sistem tidak dapat dilakukan secara menyeluruh pada tahap awal pelaksanaan magang. Hal ini disebabkan oleh keterbatasan waktu *stakeholder* serta perbedaan fokus kebutuhan antar bagian rumah sakit. Beberapa kebutuhan baru muncul setelah sistem mulai dikembangkan dan diuji coba. Kondisi tersebut mengharuskan adanya penyesuaian ulang terhadap desain dan alur sistem yang telah dibuat. Perubahan kebutuhan ini berpotensi memengaruhi jadwal pengerjaan dan konsistensi sistem. Oleh karena itu, pengelolaan perubahan kebutuhan menjadi salah satu tantangan utama selama pengembangan *website*.

3. *Appointment* sistem sebelumnya yang kurang optimal

   Pada sistem lama, proses *appointment* masih memiliki beberapa kendala yang cukup signifikan. Salah satu kendala utama adalah pasien dapat melakukan *appointment* dokter di jam berapa saja tanpa menyesuaikan

dengan jadwal praktik dokter yang tersedia. Hal ini sering menimbulkan ketidaksesuaian jadwal antara pasien dan dokter. Selain itu, pasien diwajibkan untuk melakukan *login* terlebih dahulu sebelum dapat membuat *appointment*, yang kurang efisien dan menyulitkan, terutama bagi pasien baru atau pasien yang lupa akun. Proses *appointment* pada sistem lama juga belum memiliki mekanisme konfirmasi oleh admin, sehingga data *appointment* kurang terkontrol dan berpotensi menimbulkan kesalahan. Di sisi lain, seluruh *appointment* masuk ke satu jalur komunikasi yang sama, menyebabkan penumpukan *chat* dan memperlambat proses pelayanan kepada pasien. Oleh karena itu, sistem lama dinilai kurang efektif.

3.5 Solusi Atas Kendala yang Ditemukan

Sejumlah kendala yang muncul selama pelaksanaan kerja magang menuntut adanya solusi yang sesuai agar pengembangan *website* Rumah Sakit Mentari dapat berlangsung secara optimal. Setiap kendala dianalisis untuk menentukan langkah penyelesaian yang sesuai dengan kondisi dan kebutuhan rumah sakit. Solusi yang diterapkan tidak hanya berfokus pada aspek teknis, tetapi juga pada metode kerja dan pendekatan pengembangan sistem. Dengan penerapan solusi yang tepat, diharapkan kendala yang muncul dapat diminimalkan dan tidak menghambat proses pengembangan. Solusi-solusi tersebut juga menjadi bagian dari pembelajaran selama pelaksanaan magang. Beberapa solusi yang diterapkan dalam penelitian ini meliputi.

1. Perancangan ulang sistem berbasis prototipe

   Untuk mengatasi keterbatasan dokumentasi *website* lama, pengembangan *website* dilakukan dengan merancang sistem baru secara menyeluruh. Proses perancangan diawali dengan pembuatan prototipe menggunakan *Figma*. Prototipe ini berfungsi sebagai gambaran visual dari tampilan dan alur sistem yang akan dikembangkan. Dengan adanya prototipe, proses implementasi dapat dilakukan secara lebih terarah. Prototipe juga memudahkan penyampaian ide dan konsep kepada pihak rumah sakit. Selain

itu, kesalahan desain dapat diminimalkan sebelum masuk ke tahap pengkodean.

2. Strategi Penanganan Perubahan Kebutuhan Sistem

Untuk mengatasi perubahan kebutuhan sistem yang muncul selama proses pengembangan, digunakan metode prototyping yang bersifat iteratif. Setiap fitur dikembangkan secara bertahap dan dievaluasi bersama stakeholder terkait. Masukan dari stakeholder digunakan sebagai dasar untuk melakukan perbaikan dan penyempurnaan sistem secara berkelanjutan. Pendekatan ini memungkinkan penyesuaian kebutuhan dilakukan tanpa harus mengulang seluruh proses pengembangan sehingga proses pengembangan dapat berjalan lebih efisien dan terarah. Selain itu, untuk mendukung kebutuhan konten website yang bersifat dinamis, seperti penambahan gambar dan informasi layanan, dibuat sebuah admin panel. Admin panel ini memungkinkan setiap divisi terkait, seperti divisi marketing, untuk diberikan hak akses sesuai kewenangannya. Melalui admin panel tersebut, masing-masing divisi dapat melakukan pembaruan konten website secara mandiri. Setiap perubahan yang disimpan melalui admin panel akan langsung diterapkan pada website layanan, sehingga informasi yang ditampilkan selalu sesuai dengan kebutuhan terbaru.

3. Implementasi sistem *appointment* terintegrasi *WhatsApp bussiness*

Sebagai solusi dari permasalahan tersebut, diterapkan sistem baru yang dirancang untuk meningkatkan efisiensi dan kualitas pelayanan. Pada sistem baru, pasien tidak perlu melakukan *login* untuk membuat *appointment*, sehingga proses menjadi lebih mudah dan cepat. Pasien diwajibkan memilih jadwal dokter terlebih dahulu sesuai dengan jadwal praktik yang tersedia, sehingga dapat meminimalkan terjadinya kesalahan waktu *appointment*. Setelah memilih jadwal, pasien mengisi *form appointment* yang secara otomatis akan dikirimkan ke *WhatsApp* dan selanjutnya dikonfirmasi oleh admin.

Pemilihan *WhatsApp* Business sebagai pihak ketiga dilakukan karena *WhatsApp* merupakan aplikasi komunikasi yang sudah umum digunakan oleh

masyarakat, sehingga memudahkan pasien dalam proses pendaftaran dan komunikasi. Selain itu, *WhatsApp* Business memiliki fitur yang mendukung kebutuhan pelayanan, seperti penggunaan nomor resmi, pengelolaan pesan yang lebih terstruktur, serta kemudahan dalam membalas pesan dari pasien. Dengan memanfaatkan *WhatsApp* Business, proses konfirmasi *appointment* dapat dilakukan secara cepat dan efisien tanpa memerlukan aplikasi tambahan. Selain itu, sistem baru juga membagi jalur komunikasi berdasarkan jenis layanan, yaitu *appointment* rawat jalan diarahkan ke *WhatsApp* registrasi rawat jalan, *appointment* MCU diarahkan ke *WhatsApp* MCU, dan *appointment* rawat inap diarahkan ke *WhatsApp* rawat inap. Pembagian jalur komunikasi ini bertujuan untuk menghindari penumpukan *chat* pada satu admin serta mempercepat proses pelayanan. Dengan demikian, proses konfirmasi dan pengelolaan *appointment* menjadi lebih terorganisir, efektif, dan mampu meningkatkan kualitas pelayanan kepada pasien.