BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dalam beberapa tahun terakhir membawa perubahan yang cukup signifikan di berbagai bidang, termasuk pada layanan otomotif. Digitalisasi perlahan mendorong banyak proses yang sebelumnya dikerjakan secara manual beralih ke sistem yang lebih tertata dan berbasis data. Di Indonesia, perubahan ini terlihat dari meningkatnya penggunaan teknologi digital untuk mendukung aktivitas usaha, sebagaimana disampaikan oleh (Lebang, 2023) laporan mengenai transformasi digital nasional. Teknologi kini bukan sekadar pelengkap, tetapi menjadi unsur penting agar bisnis tetap kompetitif dan mampu beroperasi secara lebih cepat serta akurat.

Meskipun demikian, kondisi di lapangan menunjukkan bahwa masih banyak bengkel, terutama yang dikelola oleh pelaku UKM, bergantung pada pencatatan manual untuk pengelolaan jadwal servis, pemesanan, dan stok suku cadang. Metode ini cukup rentan menimbulkan berbagai kendala seperti pencatatan yang tidak konsisten, kesalahan input, hingga kesulitan mencari data tertentu ketika diperlukan. Situasi tersebut berdampak langsung pada kualitas pelayanan. (Gemawaty & Yuliani, 2023)juga menegaskan bahwa penerapan sistem informasi yang terstruktur dapat membantu pemilik bengkel meningkatkan akurasi data serta mempercepat proses pelayanan kepada pelanggan.

Di sisi lain, jumlah kendaraan yang terus bertambah setiap tahun membuat pelanggan menginginkan layanan yang lebih cepat dan transparan. Kemudahan melakukan *booking* tanpa harus datang langsung, mengetahui status pengerjaan, sampai mendapatkan informasi ketersediaan layanan atau suku cadang menjadi kebutuhan yang semakin umum. Hal ini membuat keberadaan sistem yang mampu mengintegrasikan data pelanggan, jadwal servis, dan inventaris menjadi semakin penting.

Sistem informasi berbasis *web* menjadi salah satu solusi yang relevan untuk menjawab kebutuhan tersebut. Dengan satu platform, proses pencatatan, penjadwalan, dan pengelolaan inventaris dapat dilakukan secara terpusat. Pada skala yang lebih besar, pendekatan *multi-tenant* memungkinkan satu sistem digunakan oleh banyak bengkel dengan tetap menjaga pemisahan data dan aspek keamanan. (Pushpan, 2024)menjelaskan bahwa model *multi-tenant* efektif dalam mendukung pengembangan aplikasi yang skalabel serta efisien pada layanan berbasis SaaS.

Berdasarkan kondisi tersebut, diperlukan sebuah sistem yang dapat membantu proses *booking* layanan dan pengelolaan inventaris secara lebih tertib serta akurat. Selain merancang sistemnya, diperlukan pula pengujian untuk memastikan bahwa fungsi-fungsi utama berjalan sesuai kebutuhan pengguna dan bahwa performa sistem mampu mendukung aktivitas operasional bengkel secara stabil. Pengujian ini penting untuk menilai apakah sistem benar-benar layak digunakan dalam konteks operasional sehari-hari.

Penelitian ini merancang dan mengembangkan aplikasi Bengkelta, yaitu sistem informasi *booking* dan kontrol inventaris bengkel *multi-tenant* berbasis *web*. Pengembangan sistem menggunakan metode *Waterfall* dan memanfaatkan *ReactJS*

pada sisi antarmuka, *Golang* sebagai *backend service*, serta *MySQL* sebagai basis data. Selain proses pengembangan, penelitian ini juga melakukan pengujian fungsi dan kinerja sistem untuk memastikan reliabilitas dan kesesuaiannya dengan kebutuhan bengkel. Pengujian dilakukan menggunakan pendekatan *Black Box Testing*, yang berfokus pada kesesuaian *output* terhadap skenario penggunaan tanpa melihat struktur internal kode. Melalui rangkaian pengujian ini, sistem diharapkan mampu menunjukkan performa yang stabil, alur kerja yang tepat, serta mendukung operasional bengkel secara lebih terintegrasi dan responsif, sejalan dengan arah transformasi digital di sektor otomotif di Indonesia.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka dapat dirumuskan beberapa permasalahan yang akan dibahas dalam penelitian ini sebagai berikut:

1. Bagaimana merancang dan membangun sistem informasi berbasis *web* untuk melakukan *booking* layanan dan kontrol inventaris bengkel secara terintegrasi?
2. Bagaimana menguji fungsi dan kinerja sistem informasi berbasis *web* tersebut?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah yang telah disebutkan, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang dan membangun aplikasi Bengkelta, yaitu sistem informasi *booking* dan kontrol inventaris bengkel *multi-tenant* berbasis *web* menggunakan *Golang* sebagai *backend*, *ReactJS* sebagai *frontend*, dan *MySQL* sebagai basis data.
2. Menguji sistem informasi yang dikembangkan menggunakan metode *Black Box Testing* guna memastikan bahwa sistem berfungsi dengan baik.

1.4 Batasan masalah

Berdasarkan rumusan masalah yang telah disebutkan, maka batasan dari penelitian ini adalah sebagai berikut:

1. Sistem belum menerapkan fitur pembayaran *online* secara langsung dan tidak terintegrasi dengan *payment gateway*.
2. Sistem belum di *hosting* (*deployment*) dan masih berjalan dalam lingkungan pengembangan lokal (*local environment*).

1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini antara lain sebagai berikut:

1. Memberikan solusi digital terintegrasi untuk mengelola proses *booking* layanan dan kontrol inventaris suku cadang pada sistem bengkel berbasis *multi-tenant* secara efisien dan akurat.
2. Meningkatkan efektivitas operasional melalui sistem berbasis *web* yang mampu diakses secara *real-time* oleh berbagai bengkel (*multi-tenant*).

1.6 Landasan Teori

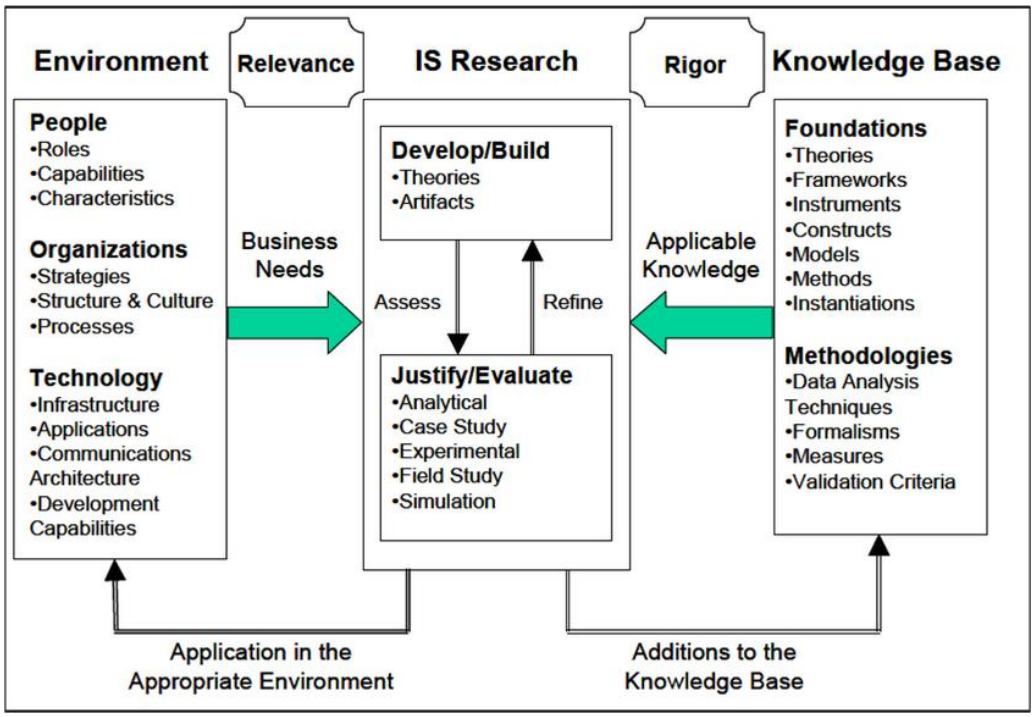
1.6.1 Sistem Informasi Berbasis *Web*

Sistem Informasi Berbasis *Web* (SIBW) merupakan sebuah kerangka kerja teknologi informasi yang dirancang untuk mengelola, memproses, dan menyajikan data melalui antarmuka yang dapat diakses menggunakan peramban (*browser*) internet. Sistem ini menjadi solusi fundamental di era digital karena menawarkan aksesibilitas *real-time* dan kemudahan akses dari berbagai lokasi tanpa terikat pada instalasi perangkat lunak di perangkat lokal (Arujisaputra, 2025).

SIBW memiliki peran krusial dalam mengatasi permasalahan operasional manual dan terfragmentasi, seperti yang dihadapi oleh bengkel UKM. Penelitian menunjukkan bahwa pengembangan sistem berbasis *web* yang terintegrasi berfungsi sebagai solusi utama untuk memusatkan pengelolaan data, mempercepat penyampaian informasi, dan mendukung pengambilan keputusan yang lebih baik (Ikhwan et al., 2025). Secara praktis, penggunaan SIBW memungkinkan Bengkelta mengintegrasikan seluruh data operasional termasuk data pelanggan, jadwal *booking*, dan inventaris ke dalam satu platform yang sama. Hal ini secara langsung berkontribusi pada peningkatan efisiensi operasional, sebab sistem digital dapat mengotomatisasi pengolahan data yang sebelumnya dilakukan secara manual, sekaligus meminimalkan kesalahan dan inefisiensi manusia. Dengan demikian, SIBW adalah fondasi utama untuk mencapai tujuan penelitian ini dalam mewujudkan pengelolaan bengkel yang terstruktur, efisien, dan modern.

1.6.2 Ruang Lingkup Penelitian Sistem Informasi

Studi ini mengadopsi kerangka *Design Science Research* (DSR) sebagai paradigma utama dalam pengembangan artefak sistem informasi, yang fokus pada penciptaan solusi inovatif untuk masalah-masalah praktis (vom Brocke et al., 2020). DSR memandang sistem informasi (SI) sebagai interaksi kompleks antara tiga elemen utama: Manusia (*People*), Organisasi (*Organization*), dan Teknologi (*Technology*). Ketiga elemen ini membentuk *Environment* yang menjadi sumber permasalahan dan tempat aplikasi diterapkan. Hubungan antara domain lingkungan, proses penelitian (*IS Researc*h), dan dasar pengetahuan (*Knowledge Base*) dapat dilihat pada Gambar 1 di bawah. Pemahaman yang komprehensif terhadap ketiga ruang lingkup ini sangat penting untuk memastikan sistem yang dikembangkan memiliki utilitas dan relevansi (Hevner et al., 2004).Manusia merujuk pada pengguna akhir, yang kebutuhannya menentukan antarmuka sistem. Organisasi adalah lingkungan bisnis (bengkel) tempat proses dan struktur bisnis diotomatisasi. Sementara itu, Teknologi adalah komponen fisik dan perangkat lunak yang mendukung fungsionalitas sistem (misalnya, *ReactJS*, *Golang*, dan *MySQL*). Dengan menganalisis gap dan kebutuhan di antara ketiga domain ini (kurangnya sistem yang mendukung *multi-tenant* di lingkungan organisasi bengkel), penelitian ini memastikan bahwa solusi Bengkelta yang dibangun tidak hanya berfungsi secara teknis, tetapi juga secara efektif menyelesaikan masalah efisiensi operasional bengkel.



Gambar 1 Kerangka penelitian sistem informasi

1.6.3 Transformasi Digital dan Efisiensi Organisasi

Transformasi digital didefinisikan sebagai adopsi teknologi digital secara menyeluruh dalam proses bisnis untuk meningkatkan kinerja dan mencapai nilai baru (Syahruddin, 2023). Bagi Usaha Mikro, Kecil, dan Menengah (UMKM) sektor otomotif seperti bengkel, transformasi ini bukan lagi pilihan, melainkan keharusan strategis. Tujuannya adalah untuk meningkatkan efisiensi operasional, yang merupakan indikator utama keberhasilan bisnis. Efisiensi operasional diukur dari kemampuan organisasi untuk menghasilkan output yang maksimal dengan sumber daya (input) yang minimal, termasuk meminimalkan waktu tunggu dan kesalahan pencatatan.

Penerapan teknologi digital terbukti memberikan dampak positif signifikan. Sistem informasi berbasis *web*, seperti yang diterapkan dalam penelitian ini, dapat mengurangi proses manual, mempercepat alur kerja, dan mendukung pengambilan keputusan yang akurat (Saleh et al., 2023). Secara spesifik, digitalisasi layanan otomotif(Kusuma & Prasetyo, 2023)memungkinkan bengkel mengelola *booking*, inventaris suku cadang, dan riwayat pelanggan secara terpusat, yang pada akhirnya meningkatkan kualitas layanan purna jual dan daya saing. Digitalisasi ini mendorong kematangan digital bengkel. Kematangan digital adalah tahapan di mana organisasi telah mampu memanfaatkan teknologi digital secara efektif untuk mengoptimalkan proses bisnis inti (Syahruddin, 2023). Dengan demikian, pengembangan Bengkelta merupakan upaya nyata untuk mencapai pengelolaan data terintegrasi sebagai kunci untuk meningkatkan efisiensi operasional secara berkelanjutan di sektor jasa otomotif.

1.6.4 Konsep Fungsionalitas Aplikasi

1.6.4.1 Konsep *Booking* Layanan Digital

Layanan *booking* digital (*online booking*) adalah proses di mana pelanggan dapat menjadwalkan layanan atau janji temu secara mandiri melalui platform berbasis *web* tanpa intervensi manusia secara langsung. Konsep ini krusial untuk manajemen waktu dan sumber daya dalam bisnis jasa. Secara teoretis, implementasi sistem *booking* digital bertujuan untuk meningkatkan efisiensi operasional bengkel dengan mengeliminasi penumpukan antrian tak terduga (*walk-in*) dan memfasilitasi penjadwalan beban kerja yang merata untuk mekanik (Kusuma & Prasetyo, 2023). Fungsionalitas ini juga harus memiliki integrasi data yang kuat, di mana sistem berbasis *web* yang terintegrasi penting untuk memastikan bahwa semua pihak (pelanggan dan manajemen) memiliki visibilitas yang sama dan data yang akurat mengenai ketersediaan jadwal dan status layanan (Ichsan & Setiawati, 2025). Fungsionalitas booking digital ini meningkatkan kepuasan pelanggan karena adanya kepastian waktu layanan dan meminimalkan waktu tunggu (*waiting time*).

1.6.4.2 Kontrol Inventaris Suku Cadang

Kontrol inventaris merujuk pada mekanisme pengelolaan persediaan barang (suku cadang) yang bertujuan untuk menjaga keseimbangan antara ketersediaan stok yang optimal dan biaya penyimpanan yang minimal. Sistem kontrol inventaris digital sangat penting untuk mendukung manajemen *procurement* dan kelancaran bisnis, memastikan proses servis tidak tertunda karena kekurangan suku cadang (Christian & Fajriah, 2020). Sistem informasi inventaris berbasis *web* memiliki fungsi utama untuk meminimalkan kesalahan dalam pengelolaan persediaan, memantau stok secara *real-time*, dan mendukung pembuatan laporan yang akurat (Zhafira et al., 2025). Dengan sistem ini, masalah kehabisan stok (*stockout*) yang dapat menunda layanan dan kelebihan stok (*overstock*) yang meningkatkan biaya penyimpanan dapat dihindari. Fungsionalitas ini memastikan bahwa suku cadang selalu tersedia saat diperlukan, yang berkontribusi langsung pada kelancaran alur kerja operasional dan mendukung efisiensi waktu layanan secara keseluruhan.

1.6.5 Arsitektur *Multi-Tenant*

Arsitektur *Multi-Tenant* adalah model arsitektur perangkat lunak di mana satu *instance* (salinan) dari aplikasi perangkat lunak dan infrastruktur pendukungnya melayani beberapa organisasi atau entitas pelanggan yang disebut *tenant*. Dalam konteks sistem Bengkelta, *multi-tenant* berarti satu platform aplikasi yang sama digunakan oleh berbagai bengkel yang berbeda secara bersamaan. Arsitektur ini adalah kerangka kerja yang komprehensif untuk membangun aplikasi *Software-as-a-Service* (SaaS) yang berskala (*scalable*) (Pushpan, 2024).

Penerapan *multi-tenant* menawarkan beberapa keunggulan strategis. Dari sudut pandang operasional dan biaya, model ini terbukti dapat menghemat sumber daya infrastruktur secara signifikan dan mempermudah pemeliharaan sistem, karena *update* dan *patch* keamanan hanya perlu diterapkan satu kali untuk melayani semua *tenant* (Pushpan, 2024). Aspek kritis dari arsitektur ini adalah isolasi data. Meskipun semua *tenant* berbagi instance aplikasi dan database yang sama (biasanya dalam satu skema

atau table yang dipartisi), sistem harus menjamin bahwa data milik satu bengkel tidak dapat diakses atau dilihat oleh bengkel lain. Isolasi data ini penting untuk menjaga keamanan dan kerahasiaan data masing-masing organisasi. Dengan mengadopsi arsitektur *multi-tenant*, penelitian ini bertujuan untuk menyediakan solusi digital yang efisien dan hemat biaya bagi banyak bengkel secara simultan, sehingga memperluas dampak positif transformasi digital.

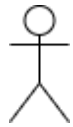
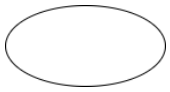
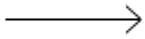
1.6.6 Pemodelan Sistem Berbasis UML

Unified Modeling Language (UML) adalah bahasa standar yang digunakan untuk memvisualisasikan, menspesifikasikan, membangun, dan mendokumentasikan artefak dari sistem berorientasi objek (Pressman & Maxim, 2020). Tiga diagram utama UML digunakan dalam penelitian ini untuk memodelkan sistem Bengkelta.

1.6.6.1 *Use Case Diagram*

Use Case Diagram adalah salah satu pemodelan untuk perilaku (behavior) sistem yang akan dibuat, di mana ia menguraikan interaksi antara satu atau lebih aktor dengan sistem. Diagram ini menggambarkan urutan interaksi yang saling berkaitan antara sistem dan aktor dari perspektif pengguna (*external view*). Tujuannya adalah mendefinisikan batas-batas sistem dan mengorganisasi persyaratan fungsional, serta mengetahui fungsi apa saja yang ada di dalam sistem dan aktor mana yang berhak menggunakan fungsi-fungsi tersebut.

Tabel 1 Komponen *use case diagram*

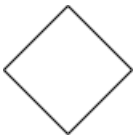
SIMBOL	NAMA	KETERANGAN
	*Actor*	Mewakili peran orang, sistem yang lain, atau alat ketika berinteraksi dengan *use case*
	*Use Case*	Abstraksi dan interaksi antara sistem dan aktor
	*Association*	Abstraksi dari penghubung antara aktor dengan *use case*
	*Generalization*	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*
<<include>>	*Include*	Menunjukkan bahwa suatu *use case* tambahan merupakan fungsionalitas

		dari *use case* lainnya jika suatu kondisi terpenuhi

1.6.6.2 *Activity Diagram*

Activity Diagram adalah pemodelan yang dilakukan pada suatu sistem untuk menggambarkan aktivitas sistem berjalan dan merepresentasikan aliran proses atau alur kerja (*workflow*). Diagram ini sangat mirip dengan *flowchart* karena memodelkan alur kerja dari satu aktivitas ke aktivitas lainnya atau dari aktivitas ke status, termasuk di dalamnya keputusan-keputusan yang mungkin terjadi. Diagram ini memodelkan proses bisnis dan urutan aktivitas dalam sebuah proses.

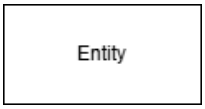
Tabel 2 Komponen pada *activity diagram*

SIMBOL	NAMA	KETERANGAN
	Start Point	Titik awal bagaimana objek diawali atau dibentuk
	End Point	Titik akhir objek
	Activities	Memperlihatkan bagaimana masing masing antarmuka kelas saling berinteraksi satu sama lain
	Decision	Menggambarkan suatu Keputusan atau tindakan yang harus diambil pada kondisi tertentu

1.6.6.3 *Entity Relationship Diagram*

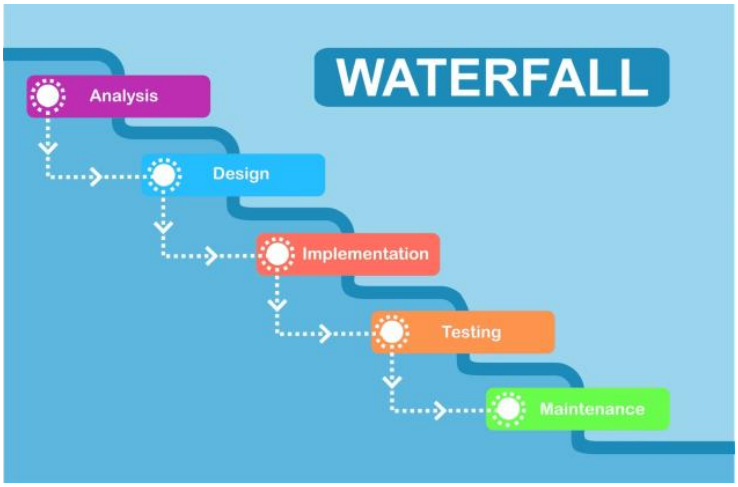
Entity Relationship Diagram (ERD) adalah diagram yang berbentuk notasi grafis yang merupakan salah satu alat utama dalam perancangan basis data. ERD memodelkan struktur data secara konseptual, yang mendeskripsikan hubungan antara data-data yang saling berhubungan. ERD adalah tahap dasar dalam membuat database dan merupakan teknik perancangan yang paling banyak digunakan karena semua entitas, atribut, dan relasinya harus dirancang secara lengkap dan detail.

Tabel 3 Komponen pada *entity relationship diagram*

SIMBOL	NAMA	KETERANGAN
	Entity	Merupakan suatu simbol untuk mewakili suatu objek dengan karakteristik sama yang dilengkapi oleh atribut.
	Attribute	Merupakan suatu simbol yang menjelaskan suatu entitas karakteristik dan juga relasinya.
	Strong Relationship	Menggambarkan hubungan beberapa entitas berdasarkan fakta pada suatu lingkungan.
	Connection	Menggambarkan keterkaitan antara simbol berupa garis penghubung

1.6.7 Metodologi Pengembangan *Waterfall*

Metode *Waterfall* (atau model sekuensial linear) adalah metodologi pengembangan perangkat lunak yang sistematis dan berurutan. Metode ini membagi proses pengembangan menjadi beberapa tahapan yang ketat, di mana setiap fase harus diselesaikan sepenuhnya dan divalidasi sebelum pindah ke fase berikutnya, mengalir seperti air terjun(Pressman & Maxim, 2020).



Gambar 2 Tahapan dari metode *waterfall*

1. *Requirement Analysis* (Analisis Kebutuhan) yaitu pengumpulan dan analisis persyaratan untuk mengetahui dan memahami komponen perangkat lunak yang memenuhi kebutuhan pengguna.
2. *Design* (Desain) Proses perancangan pembuatan perangkat lunak, termasuk arsitektur perangkat lunak dan representasi antarmuka perangkat lunak
3. *Implementation* (Implementasi) Proses implementasi suatu perancangan pada perangkat lunak berupa program komputer yang saling terintegrasi, sesuai dengan perancangan yang dibuat pada tahap sebelumnya.
4. *Testing* (Pengujian) Proses pengujian bertujuan untuk memverifikasi apakah sistem yang dihasilkan memenuhi kebutuhan pengguna dan meminimalkan kesalahan yang ada.
5. *Maintenance* (Pemeliharaan) Proses pemeliharaan dapat berupa pembaruan perangkat lunak ketika beberapa komponen baru dirasa perlu, atau proses perbaikan *bug* yang tidak terdeteksi pada proses pengujian sebelumnya.

1.6.8 Teknologi Pengembangan

1.6.8.1 *ReactJS*

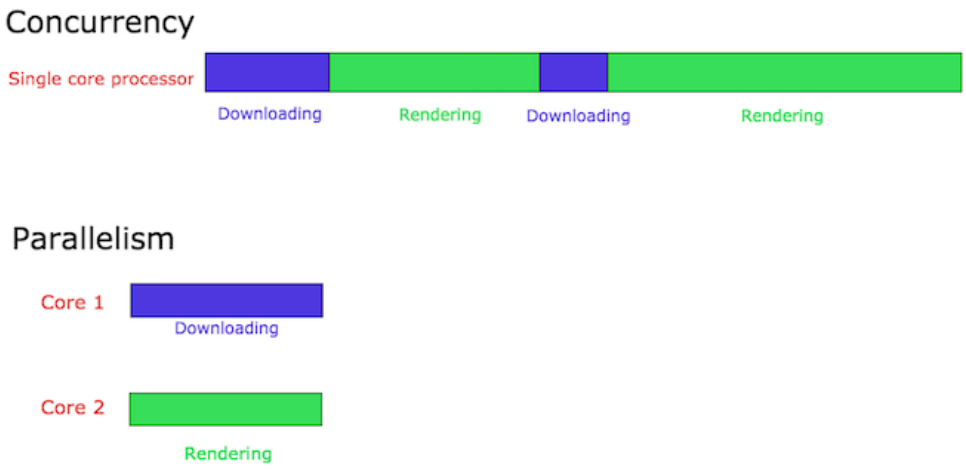
ReactJS adalah pustaka *JavaScript* sumber terbuka (*open-source*) yang digunakan untuk mengembangkan komponen antarmuka pengguna (*User Interface*/UI) yang dapat digunakan kembali (*reusable*). Teknologi ini sangat populer karena kemampuan desainnya yang deklaratif dan performa yang baik dalam pengembangan web modern (Kim et al., 2020).

ReactJS sering digunakan untuk membangun *Single Page Application* (SPA), yaitu aplikasi *web* yang hanya memuat satu halaman HTML saja dan memperbarui konten secara dinamis melalui *JavaScript*. Pendekatan SPA ini menghasilkan kinerja yang lebih cepat dan pengalaman pengguna yang mulus tanpa perlu memuat ulang halaman secara keseluruhan (Santoso, 2021).

Salah satu keunggulan utama *ReactJS* adalah implementasi *Virtual* DOM yang memungkinkan optimalisasi proses *rendering* dengan hanya memperbarui bagian-bagian spesifik dari antarmuka yang mengalami perubahan, sehingga secara signifikan mengurangi waktu *rendering* dan meningkatkan responsivitas aplikasi (Veeri, 2024).

1.6.8.2 *Golang*

Golang (atau *Go*) merupakan bahasa pemrograman yang dirancang oleh *Google* untuk efisiensi, keandalan, dan kemudahan dalam penulisan kode, khususnya untuk membangun sistem *server-side*. Bahasa ini menawarkan kinerja tinggi karena kodenya dikompilasi langsung ke kode mesin (*machine code*), sehingga proses eksekusi aplikasi menjadi sangat cepat dan stabil, ideal untuk layanan *real-time* dan sistem dengan trafik tinggi (ITGID, 2025).



Gambar 3 *Concurrency* dan *parallelism*

Kekuatan utama *Golang* terletak pada dukungan konkurensi (*concurrency*), yaitu kemampuannya menangani banyak proses secara bersamaan melalui fitur *Goroutine*. Fitur ini sangat ringan, hanya menggunakan memori sekitar 2KB per proses, sehingga memungkinkan aplikasi menjalankan ribuan proses paralel secara efisien dan mendukung skalabilitas yang baik (Skola, 2024).

Dalam eksperimen perbandingan performa, *Golang* menunjukkan hasil yang superior dibandingkan *Java* dalam aspek waktu kompilasi dan pengelolaan konkurensi, dengan kode *Go* yang lebih mudah dalam implementasi pemrograman konkuren (Togashi & Klyuev, 2014).

1.6.8.3 *MySQL*

MySQL adalah salah satu perangkat lunak sistem manajemen basis data relasional (*Relational Database Management System*/RDBMS) sumber terbuka yang paling banyak digunakan di industri untuk pengolahan data. Sebagai sebuah RDBMS, peran utama *MySQL* adalah mengatur dan mengelola data dalam bentuk tabel-tabel yang terpisah, serta menggunakan batasan seperti Kunci Primer (*Primary Key*) dan Kunci Asing (*Foreign Key*) untuk menjaga akurasi, konsistensi, dan integritas data (Pratama & Suhada, 2025).

Untuk berinteraksi dengan database server dan melakukan pengelolaan data, *MySQL* menggunakan *Structured Query Language* (SQL) sebagai bahasa database interaktif standar. SQL memungkinkan pengguna untuk melakukan berbagai fungsi penting, termasuk pendefinisian skema, serta operasi manipulasi data seperti *CREATE,*

READ, UPDATE, dan *DELETE* (CRUD), menjadikannya solusi populer untuk mengelola dan menyimpan data dengan mudah (Malahayati et al., 2023).

Performa *MySQL* sangat dipengaruhi oleh desain struktur database dan optimasi *query*, dimana penggunaan *indexing* yang tepat dapat mempercepat operasi *SELECT* namun perlu dipertimbangkan dengan cermat karena terlalu banyak *index* dapat memperlambat operasi *INSERT* dan *UPDATE* (Győrödi et al., 2021).

1.6.9 *Black Box Testing*

Black Box Testing adalah sebuah metode pengujian perangkat lunak yang fokus pada validasi fungsionalitas aplikasi dari sudut pandang pengguna, tanpa melihat struktur internal, desain kode, atau implementasi di dalamnya. Pengujian ini merujuk pada spesifikasi fungsional dan kebutuhan pengguna untuk memastikan perangkat lunak bekerja sesuai yang diharapkan, menjadikannya juga dikenal sebagai *functional testing* atau *specification-based testing* (Akademik, 2025).

Tujuan utama dari penerapan *Black Box Testing* adalah untuk memverifikasi fungsionalitas sistem secara keseluruhan dan mengidentifikasi *bug* atau kesalahan yang mungkin terlewatkan selama tahap pengembangan. Pengujian ini sangat efektif dalam menemukan kesalahan yang berkaitan dengan fungsi sistem, antarmuka pengguna, dan masalah kegunaan (*usability*) sebelum aplikasi dirilis ke publik (ITBOX, 2023).

Metode *Black Box Testing* tetap menjadi pendekatan fundamental dalam *software quality assurance* karena kemampuannya menganalisis aplikasi dari perspektif pengguna akhir dan mengevaluasi apakah *software* memenuhi *requirement* yang telah ditetapkan tanpa memerlukan akses ke *source code* (Akhtar, 2025).

BAB II
METODE PENELITIAN

2.1 Waktu dan Lokasi Penelitian

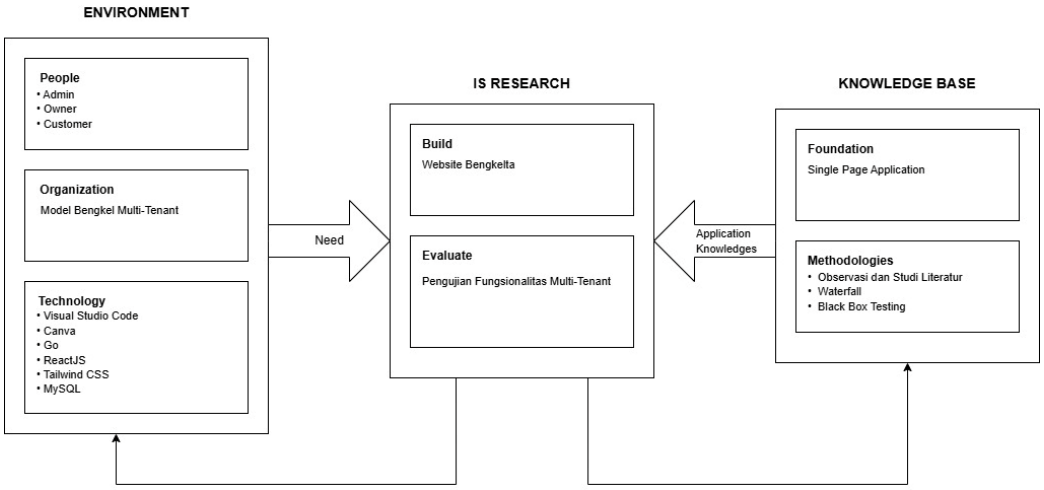
Penelitian ini dilaksanakan pada bulan Juli 2025 sampai bulan Oktober 2025 di Kota Makassar

Tabel 4 Jadwal kegiatan penelitian

No	Tahapan Penelitian	Juli				Agustus				September				Oktober			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Observasi																
2	Studi Literatur																
3	Requirement																
4	Design																
5	Implementation																
6	Testing																
7	Maintenance																

2.2 *Design Science Research*

Berikut ini adalah gambaran Kerangka *Information Systems Research Framework* yang digunakan dalam penelitian ini. Konsep metode penelitian ini menggunakan *Design Science Research* (DSR) yang sering digunakan dalam Penelitian Sistem Informasi (Hevner, A et al., 2004). Berikut pada Gambar 4 (merujuk pada Gambar DSR Anda) menjelaskan kerangka penelitian sistem informasi yang digunakan pada penelitian ini, yang terdiri dari *Environment*, *IS Research*, dan *Knowledge Base*.



Gambar 4 *Information systems research framework*

Pada aspek *Environment*, terdiri dari *People* (orang), *Organization* (organisasi), dan *Technology* (teknologi). *People* (orang) mencakup peran *Admin*, *Owner*, dan *Customer*. *Organization* (organisasi) yang menjadi fokus penelitian adalah Model Bengkel Multi-Tenant, yang dipilih karena aplikasi Bengkelta dirancang untuk dapat digunakan oleh berbagai bengkel. Kemudian, *Technology* (teknologi) yang digunakan sebagai fondasi pengembangan adalah *Visual Studio Code*, *Canva*, *Go*, *ReactJS*, *Tailwind* CSS, dan *MySQL*.

Pada aspek *IS Research*, terdiri dari *Build* (membangun) dan *Evaluate* (evaluasi). Pada tahap *Build*, dilakukan pembuatan dan perancangan *Website* Bengkelta. Pada tahap *Evaluate*, dilakukan Pengujian Fungsionalitas *Multi-Tenant* yang bertujuan untuk menilai kinerja dan efektivitas artefak. Selain itu, dilakukan analisis masalah dan analisis kebutuhan sistem untuk mengetahui masalah berdasarkan informasi yang didapatkan dan menentukan kebutuhan sistem yang akan dikembangkan.

Knowledge Base yang mendukung penelitian ini terdiri dari *Foundation* (dasar) dan *Methodologies* (metodologi). *Foundation* mencakup konsep arsitektur *Single Page Application*. Kemudian, metodologi yang digunakan pada penelitian ini yaitu Observasi dan Studi Literatur sebagai metode pengumpulan data, serta model pengembangan *System Development Life Cycle* (SDLC) *Waterfall* dan metode pengujian *Black Box Testing*. Penerapan metode DSR ini diharapkan dapat berguna dalam meningkatkan

kemudahan pemilik bengkel dalam melakukan *booking* dan kontrol inventaris secara efisien.

2.3 Metode Pengumpulan Data

Dalam penelitian ini, dilakukan beberapa tahap pengumpulan data untuk mendukung pengembangan sistem yang lebih baik. Pemilihan metode pengumpulan data sangat krusial karena hasil yang diperoleh akan menjadi dasar dalam proses pengembangan. Adapun metode pengumpulan data yang digunakan dalam penelitian ini adalah sebagai berikut:

2.3.1 Observasi

Observasi merupakan metode pengumpulan data dengan cara mengamati langsung objek penelitian di lapangan. Dalam penelitian ini, observasi dilakukan untuk memahami secara mendalam kondisi dan proses bisnis yang ada pada model Bengkel *Multi-Tenant* yang menjadi objek penelitian. Proses observasi ini bertujuan untuk mengidentifikasi kebutuhan spesifik dan alur kerja yang relevan terkait sistem *booking* dan kontrol inventaris yang akan diimplementasikan pada aplikasi Bengkelta.

2.3.2 Studi Literatur

Studi literatur adalah metode pengumpulan data dengan menelaah berbagai sumber tertulis yang relevan dengan topik penelitian. Sumber-sumber ini meliputi buku, jurnal, artikel ilmiah, laporan penelitian, serta dokumen-dokumen lain yang terkait. Dalam penelitian ini, penulis melakukan studi literatur dengan mencari berbagai informasi dari jurnal, buku, dan *website* yang relevan mengenai pengembangan aplikasi *web multi-tenant*, sistem *booking* daring, kontrol inventaris, serta teknologi pendukung seperti *Go*, *ReactJS*, dan *MySQL*.

2.4 Metode Pengembangan Sistem

Dalam penelitian ini, metode pengembangan sistem yang digunakan adalah metode *Waterfall*. Metode ini terdiri dari beberapa tahapan yang harus dilalui secara berurutan, sehingga memberikan kerangka kerja yang terstruktur dan disiplin untuk pengembangan aplikasi Bengkelta. Proses *Waterfall* yang diterapkan dalam pembangunan *Website* Bengkelta adalah sebagai berikut:

2.4.1 *Requirements* (Kebutuhan)

Pada tahapan ini, peneliti melakukan analisis untuk menentukan kebutuhan fungsional dan non-fungsional apa saja yang harus ada dalam sistem. Hal ini bertujuan untuk memahami dan mendefinisikan seluruh fitur dan alur kerja yang diharapkan pada aplikasi Bengkelta. Kebutuhan sistem didapatkan melalui kombinasi Observasi pada model bengkel dan Studi Literatur terkait sistem booking dan inventarisasi *multi-tenant*.

2.4.2 *Design* (Desain)

Pada tahapan desain, peneliti merancang diagram *use case* dan diagram aktivitas untuk memberikan gambaran umum mengenai bagaimana sistem akan bekerja. Selanjutnya, peneliti juga membuat *Entity Relationship Diagram* (ERD) untuk memvisualisasikan

struktur data dan aliran informasi data dalam sistem. Selain itu, dilakukan penyusunan rancangan antarmuka pengguna (*user interface*) untuk memberikan gambaran visual yang lebih jelas mengenai tampilan aplikasi Bengkelta.

2.4.3 *Implementation* (Implementasi

Pada tahapan ini, peneliti mulai mengimplementasikan desain yang telah dibuat. Aplikasi Bengkelta dibuat berbasis web dengan menggunakan *Go* sebagai *framework* utama pengembangan *back-end* dan *ReactJS* sebagai *framework* utama *front-end*. Selain itu, digunakan pula *Tailwind* CSS sebagai *library* untuk tampilan, dengan *MySQL* sebagai sistem manajemen basis data.

2.4.4 *Testing* (Pengujian)

Pada tahapan ini, peneliti melakukan pengujian dengan metode *Black Box Testing* untuk memastikan bahwa sistem yang telah dibangun berfungsi dengan baik sesuai dengan spesifikasi fungsional. Pengujian ini secara spesifik berfokus pada Pengujian Fungsionalitas *Multi-Tenant* aplikasi Bengkelta.

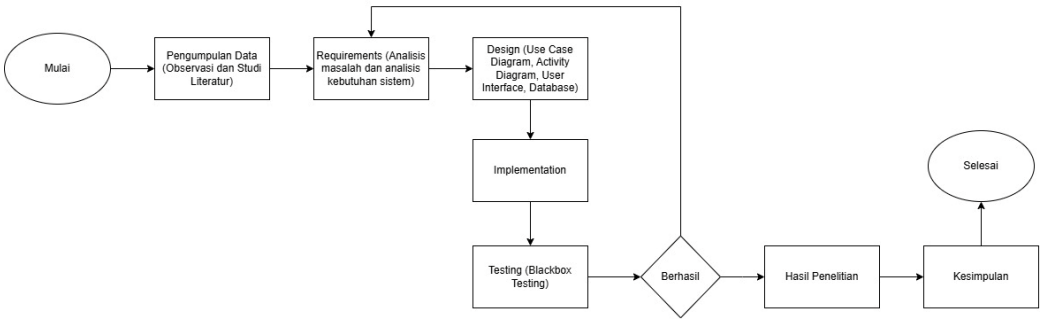
2.4.5 *Maintenance* (Pemeliharaan)

Pada tahapan ini, peneliti melakukan pemeliharaan (*maintenance*) pada aplikasi Bengkelta untuk memastikan tidak ada *bug* atau *error* yang terjadi pada *website*.

2.5 Tahapan Penelitian

Proses penelitian dimulai dengan tahap Pengumpulan Data yang meliputi observasi dan studi literatur untuk mengidentifikasi kebutuhan sistem informasi *multi-tenant*. Data yang diperoleh akan menjadi dasar untuk pengembangan sistem selanjutnya. Tahapan pengembangan sistem ini menggunakan metode *Waterfall*, yang dimulai dengan tahap *Requirements* (Analisis Kebutuhan). Pada tahap ini, peneliti menganalisis kebutuhan sistem dan alur-alur yang harus ada untuk aplikasi Bengkelta, sekaligus melakukan analisis masalah dan analisis kebutuhan sistem. Setelah itu, penelitian dilanjutkan dengan tahapan *Design* (Desain), yaitu melalui perancangan *use case diagram, activity diagram, user interface* (UI), serta perancangan *database* (ERD) untuk memberikan gambaran tentang bagaimana sistem Bengkelta akan berfungsi dan memiliki struktur data.

Tahap berikutnya adalah *Implementation* (Implementasi), yang merupakan perwujudan rancangan yang telah dibuat menjadi aplikasi berbasis *web* menggunakan teknologi *Go* dan *ReactJS*. Setelah aplikasi dibangun, *Testing* (Pengujian) dilakukan dengan menggunakan metode *Black Box Testing* untuk memastikan aplikasi Bengkelta berfungsi dengan baik sesuai harapan dan spesifikasi fungsionalitas *multi-tenant*. Jika sistem tidak memenuhi fungsi atau kebutuhan yang ditentukan, maka tahapan akan diulang kembali ke tahap *Requirements* untuk diperbaiki (disebut siklus umpan balik). Namun, jika sistem sudah berjalan sesuai dengan kebutuhan, maka penelitian dapat dianggap selesai, menghasilkan Hasil Penelitian dan Kesimpulan.



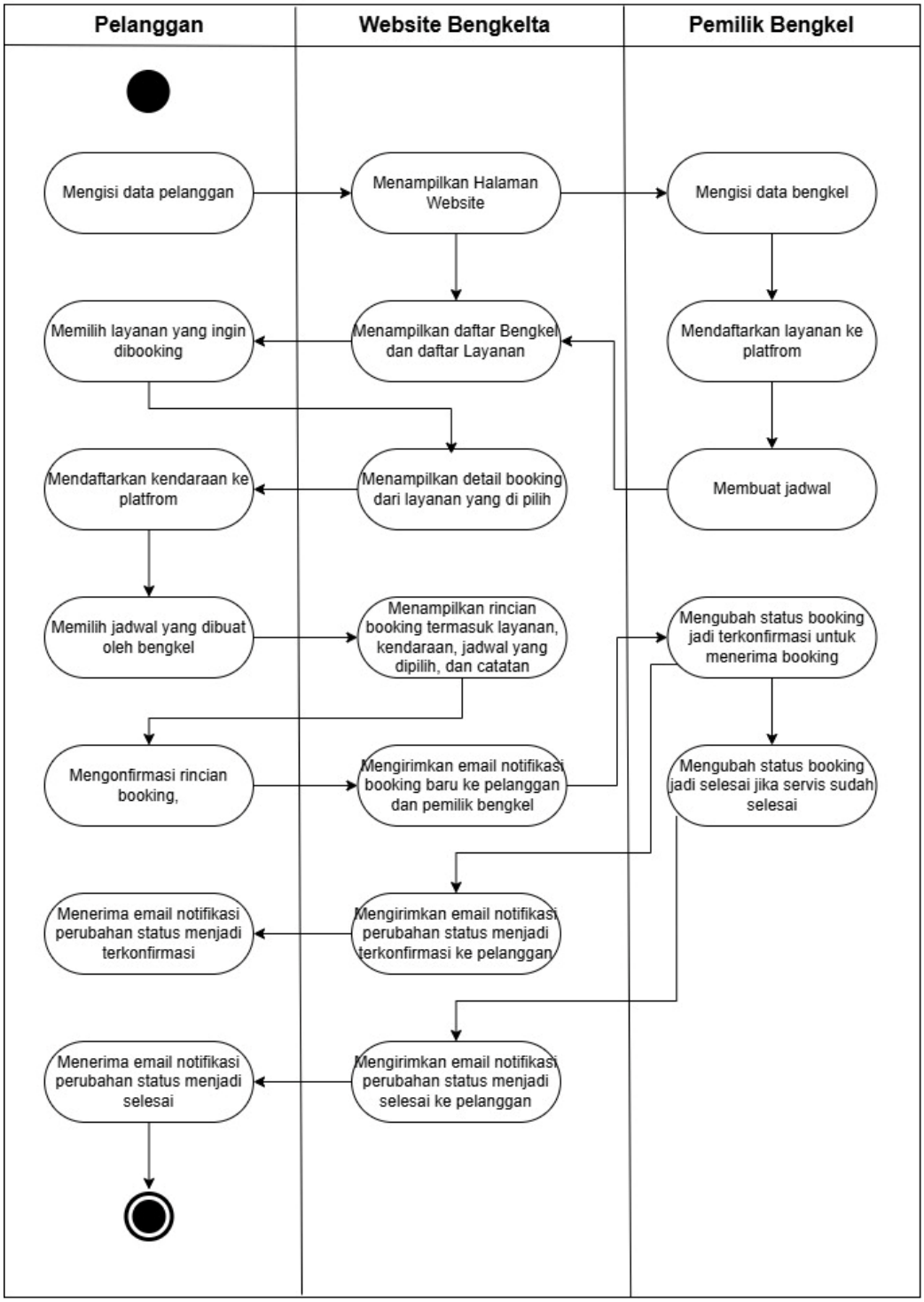
Gambar 5 Tahapan penelitian

2.6 Analisis Pengembangan Sistem

2.6.1 Analisis Masalah

Berdasarkan hasil observasi dan studi pendahuluan, ditemukan beberapa permasalahan yang terjadi pada model bengkel Usaha Kecil dan Menengah (UKM) yang beroperasi secara manual di lingkungan sektor jasa otomotif. Salah satu masalah utama adalah sulitnya pengelolaan operasional karena proses-proses penting seperti pencatatan jadwal servis, pemesanan layanan (booking), dan pengelolaan inventaris suku cadang sering kali dilakukan secara tradisional. Sistem manual ini menimbulkan berbagai permasalahan klasik, seperti duplikasi dan kesalahan pencatatan data, keterlambatan pelayanan, serta kesulitan bagi pemilik bengkel dalam memantau stok dan aktivitas operasional secara menyeluruh.

Masalah lainnya adalah kurangnya integrasi sistem informasi pada organisasi bisnis yang menyebabkan penurunan efektivitas manajemen serta memperlambat proses pengambilan keputusan yang strategis. Data masih disimpan dan dikelola menggunakan cara-cara manual atau terpisah, yang menyebabkan inefisiensi dalam layanan dan inkoordinasi. Oleh karena itu, pengembangan sebuah sistem berbasis *web multi-tenant* yang terpusat dan terintegrasi, yang mampu mengelola proses *booking* dan pengendalian inventaris, sangat diperlukan untuk mengatasi masalah-masalah tersebut. Sistem Bengkelta diharapkan dapat memberikan kemudahan bagi pelanggan dalam melakukan pemesanan layanan, sekaligus membantu pemilik bengkel dalam memantau stok barang dan jadwal servis secara *real-time*.



Gambar 6 Sistem penggunaan *website* bengkelta

2.6.2 Analisis Kebutuhan Sistem

Dalam penelitian ini, dibutuhkan perangkat lunak dan perangkat keras sebagai *tools* atau alat yang dapat mendukung penelitian, serta pengguna sistem sebagai bahan analisis kebutuhan dalam perancangan sistem. Adapun kebutuhan sistem dalam penelitian ini yaitu:

2.6.2.1 Perangkat Lunak

- *Linux*
- *Windows* 11
- *Draw.io*
- *Canva*
- *Visual Studio Code*
- *Go*
- *ReactJS*
- *Tailwind CSS*
- *MySQL*
- *Dbeaver*
- *Postman*
- *Firefox*

2.6.2.2 Perangkat Keras

Selama penelitian, peneliti menggunakan Laptop Acer Nitro V15 dengan spesifikasi *Processor* Intel core i5-13420H @ 2,1 GHz, RAM 16GB, dan SSD 1TB.

2.6.2.3 Pengguna Sistem

a. *Admin*

Admin merupakan pengguna yang berperan sebagai moderator dalam sistem. *Admin* memiliki wewenang untuk menangguhkan dan memodifikasi data pengguna, menghapus data pengguna, serta memodifikasi dan menghapus data bengkel. Selain itu, admin juga dapat memodifikasi layanan (dalam batas tertentu), menghapus layanan, mengubah status pemesanan (*booking*), dan menghapus data pemesanan.

b. *Owner*

Owner merupakan pengguna yang berperan sebagai pemilik bengkel. Pengguna dengan peran ini memiliki hak untuk memodifikasi data bengkel, menambahkan dan mengelola jadwal layanan, serta menambahkan, memodifikasi, dan menghapus data inventaris maupun layanan. Selain itu, owner juga dapat mengubah status pemesanan serta menghapus data pemesanan.

c. *Customer*

Customer merupakan peran pengguna *default* dalam sistem, yang berfungsi sebagai pengguna umum dari *website*. Pengguna dengan peran ini dapat memodifikasi data pribadinya, menambahkan, memodifikasi, dan menghapus data kendaraan miliknya sendiri, melakukan pemesanan dan pembatalan layanan (*booking*), serta mendaftar sebagai pemilik bengkel apabila diinginkan.

2.7 Perancangan Sistem

Dalam proses perancangan sistem, peneliti memanfaatkan *use case diagram* untuk merepresentasikan aktivitas yang dapat dilakukan oleh pengguna di dalam sistem. Diagram ini melibatkan tiga aktor utama dengan aktivitas atau perilaku yang berbeda, yaitu *admin, owner,* dan *customer*, sebagaimana yang ditampilkan pada Gambar 7 sebagai ilustrasi *use case diagram*.

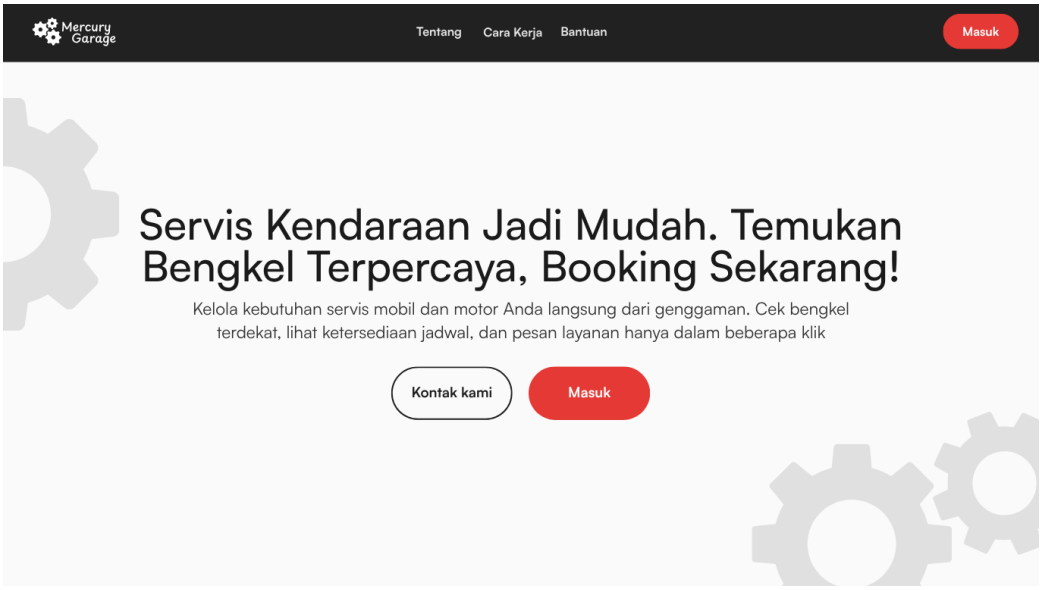
Gambar 7 *Use case diagram*

2.8 Rancangan *User Interface*

Desain *User Interface* (UI) merupakan representasi visual yang berinteraksi langsung dengan pengguna, bertujuan untuk memberikan gambaran mengenai fitur-fitur yang tersedia dalam sistem. Rancangan *User Interface* pada penelitian ini adalah sebagai berikut:

2.8.1 Halaman Utama

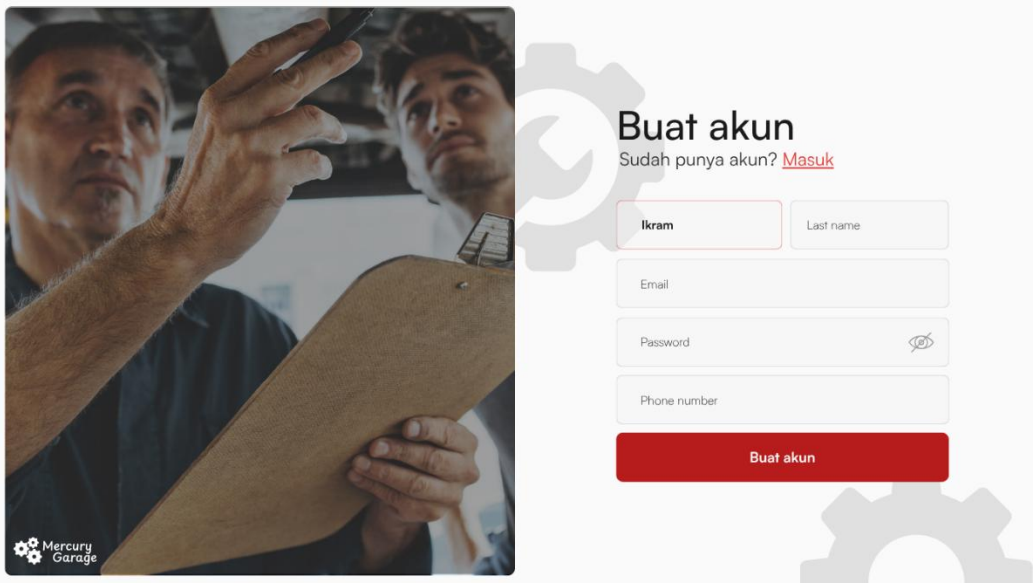
Halaman utama merupakan halaman yang diakses pertama kali ketika *website* dikunjungi



Gambar 8 Halaman utama

2.8.2 Halaman *Register*

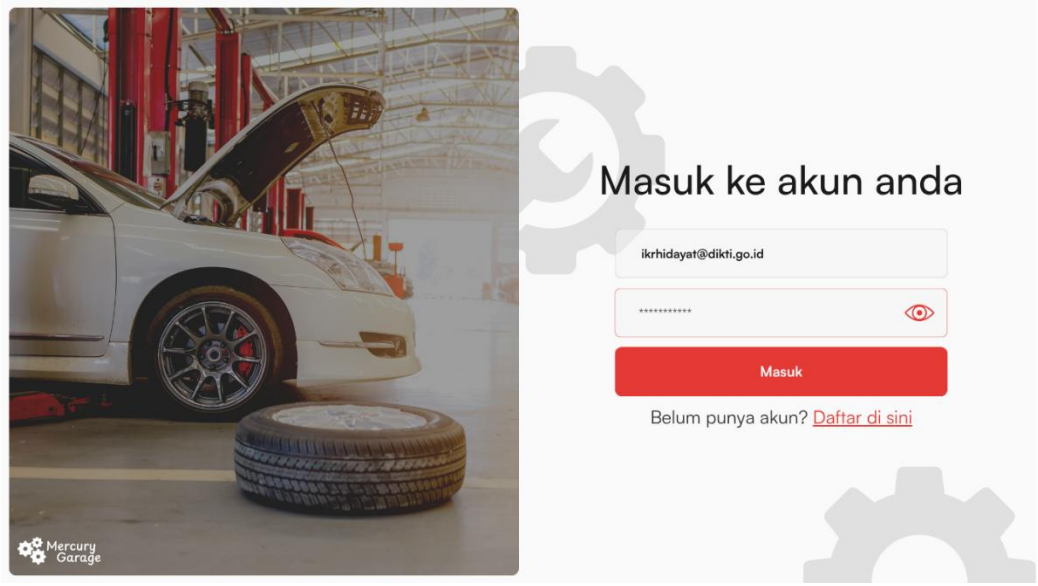
Halaman *register* merupakan halaman yang meminta pengguna untuk membuat akun yang nantinya akan di pakai untuk *login*



Gambar 9 Halaman *register*

2.8.3 Halaman *Login*

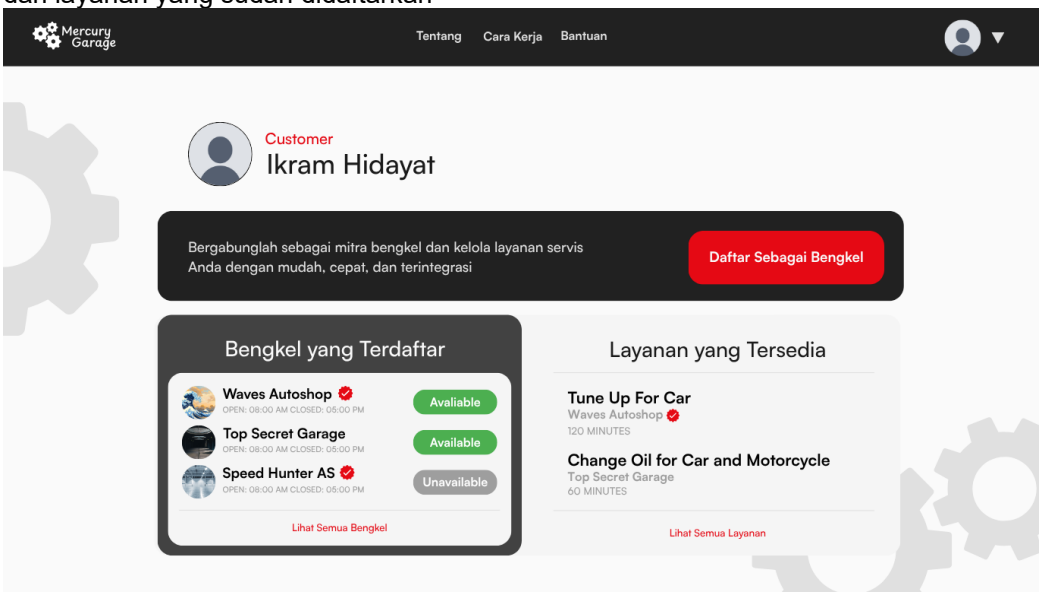
Halaman *login* merupakan halaman yang meminta pengguna untuk memasukkan *email* dan *password* sebagai langkah autentikasi sebelum dapat mengakses fitur-fitur dalam *website*



Gambar 10 Halaman *login*

2.8.4 Halaman Beranda

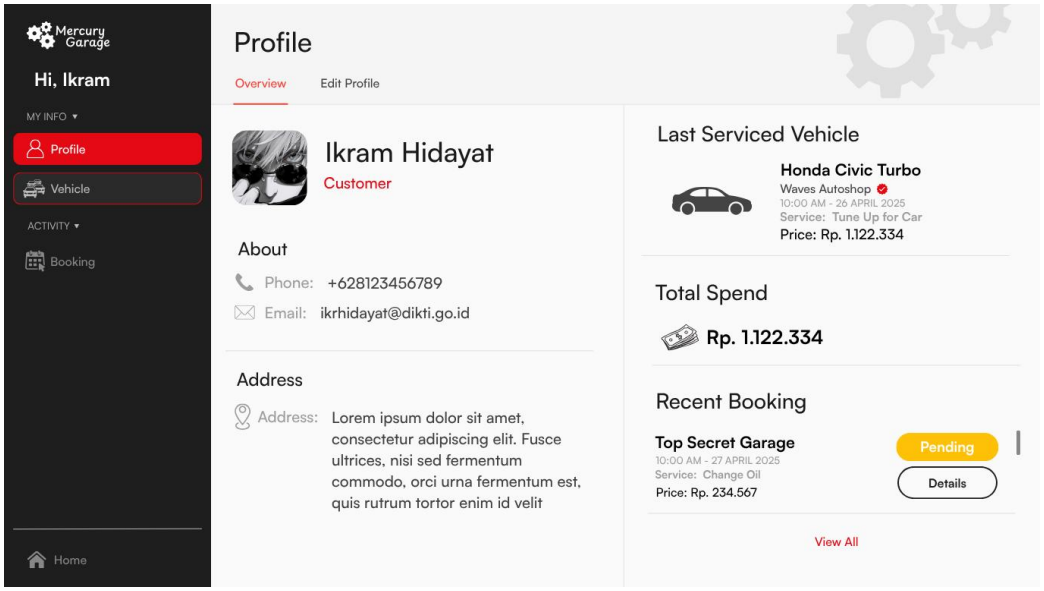
Halaman beranda menampilkan nama, *role*, CTA pendaftaran ke bengkel, *list* bengkel dan layanan yang sudah didaftarkan



Gambar 11 Halaman beranda

2.8.5 Halaman Profil

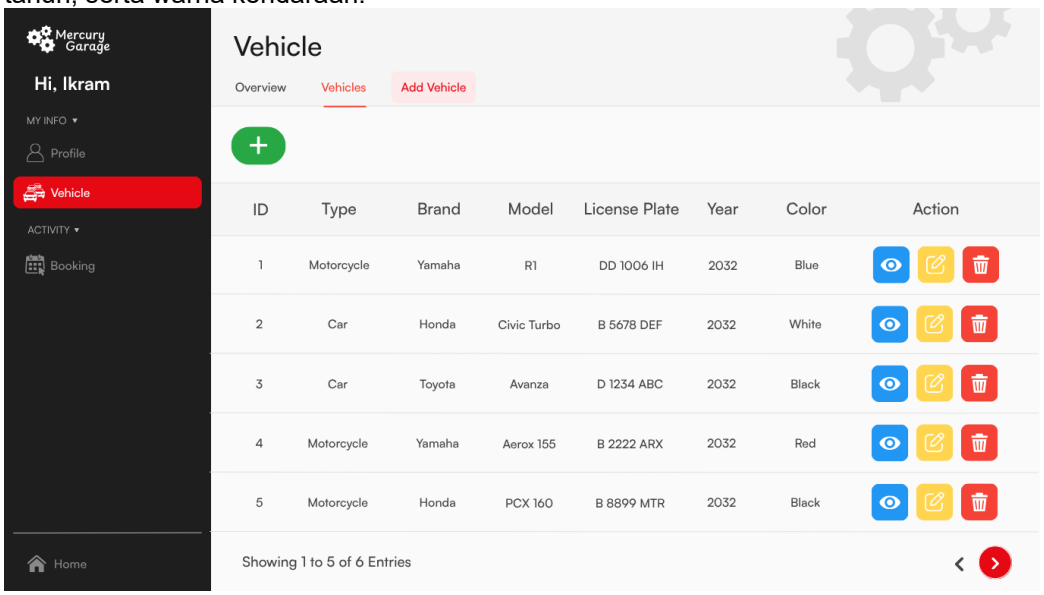
Halaman profil menampilkan ringkasan profil mulai dari nama, *role*, nomor telepon, alamat, kendaraan terakhir yang di servis, total pengeluaran, dan 2 *booking* terakhir



Gambar 12 Halaman profil

2.8.6 Halaman Manajemen Kendaraan

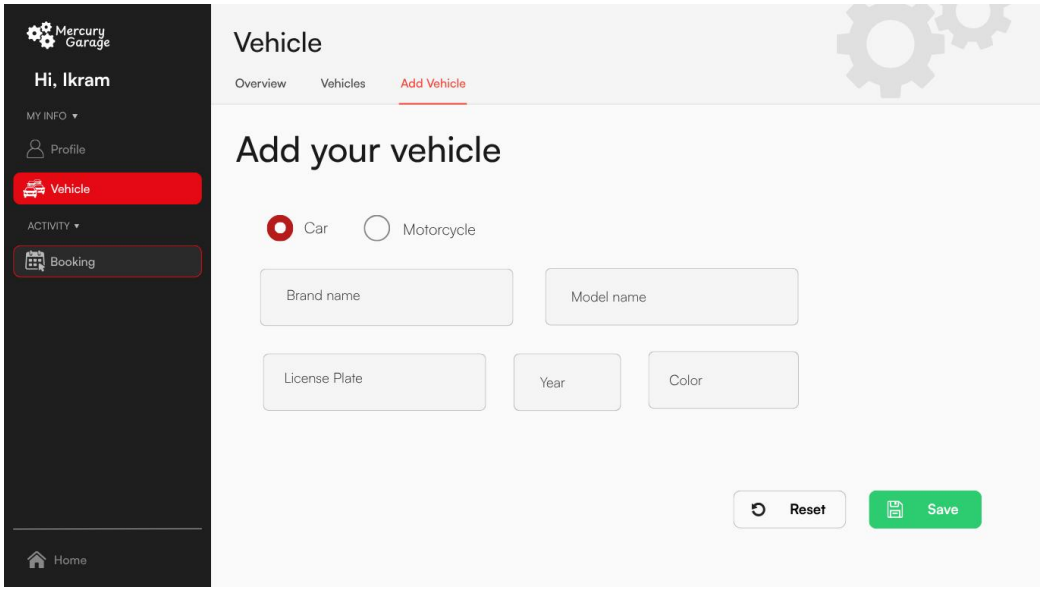
Halaman Manajemen Kendaraan berfungsi untuk menampilkan dan mengelola daftar kendaraan yang dimiliki oleh pengguna. Pada halaman ini, pengguna dapat melihat informasi kendaraan secara rinci seperti tipe kendaraan, merek, model, nomor plat, tahun, serta warna kendaraan.



Gambar 13 Halaman kendaraan

2.8.7 Halaman Tambah Kendaraan

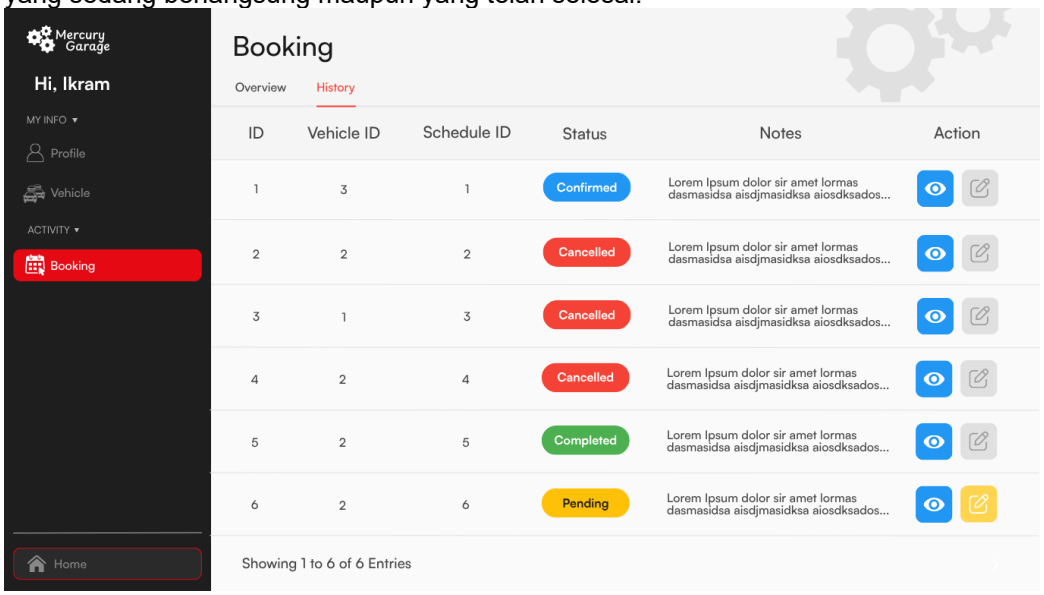
Halaman Tambah Kendaraan (*Add Vehicle*) digunakan oleh pengguna untuk menambahkan data kendaraan baru ke dalam sistem. Pada halaman ini, pengguna dapat memilih jenis kendaraan, yaitu Mobil (*Car*) atau Motor (*Motorcycle*), dengan menandai opsi yang tersedia.



Gambar 14 Halaman daftar kendaraan

2.8.8 Halaman Riwayat *Booking*

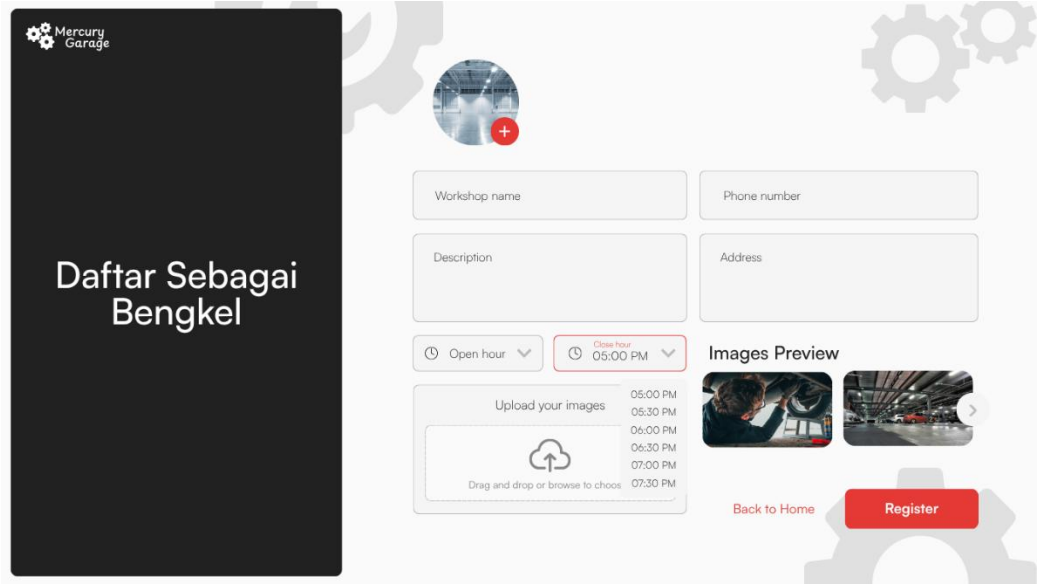
Halaman Riwayat *Booking* (*Booking History*) berfungsi untuk menampilkan daftar pemesanan servis kendaraan yang telah dilakukan oleh pengguna. Melalui halaman ini, pengguna dapat memantau status dari setiap pemesanan yang telah dilakukan, baik yang sedang berlangsung maupun yang telah selesai.



Gambar 15 Halaman riwayat *booking*

2.8.9 Halaman Registrasi Bengkel

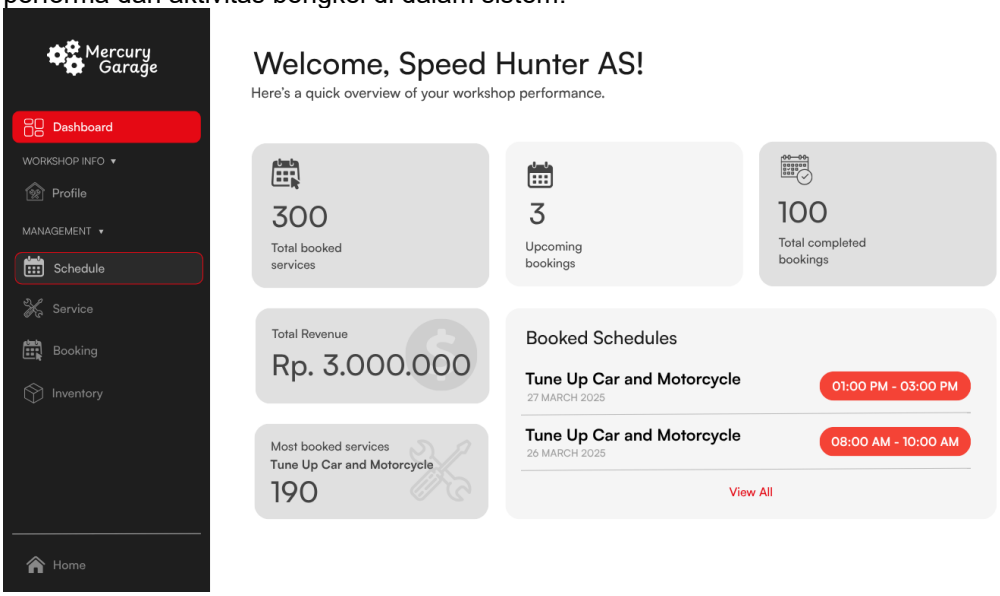
Halaman Registrasi Bengkel digunakan oleh pengguna untuk melakukan pendaftaran sebagai pemilik bengkel dalam sistem. Melalui halaman ini, pengguna dapat memasukkan berbagai informasi terkait bengkel yang akan didaftarkan agar dapat terdaftar dan dikelola di platform.



Gambar 16 Halaman registrasi bengkel

2.8.10 Halaman *Dashboard* Bengkel

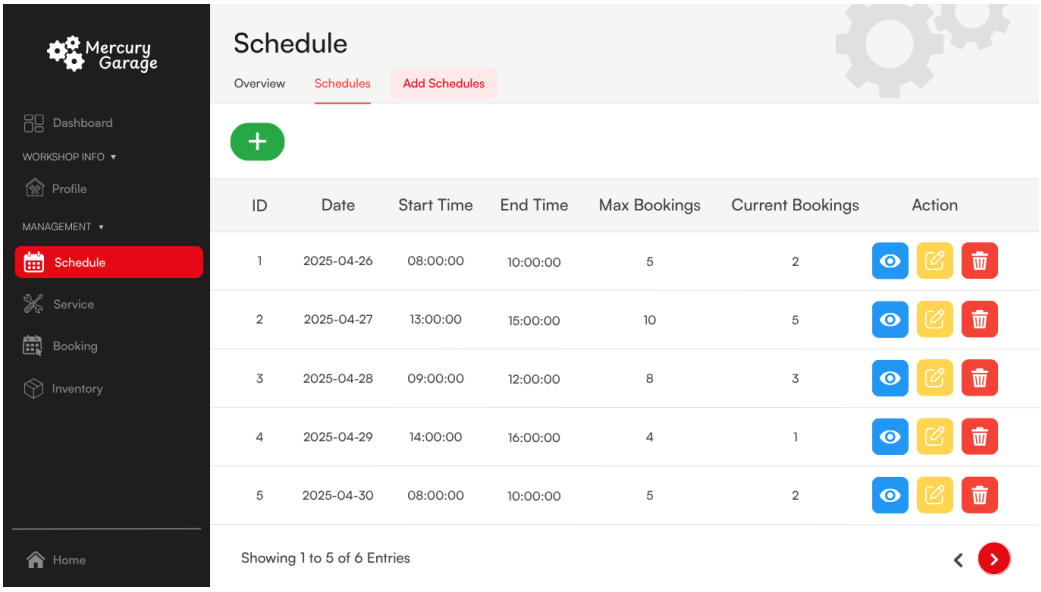
Halaman ini berfungsi untuk memberikan ringkasan atau gambaran umum terkait performa dan aktivitas bengkel di dalam sistem.



Gambar 17 *Halaman dashboard* bengkel

2.8.11 Halaman Manajemen Jadwal

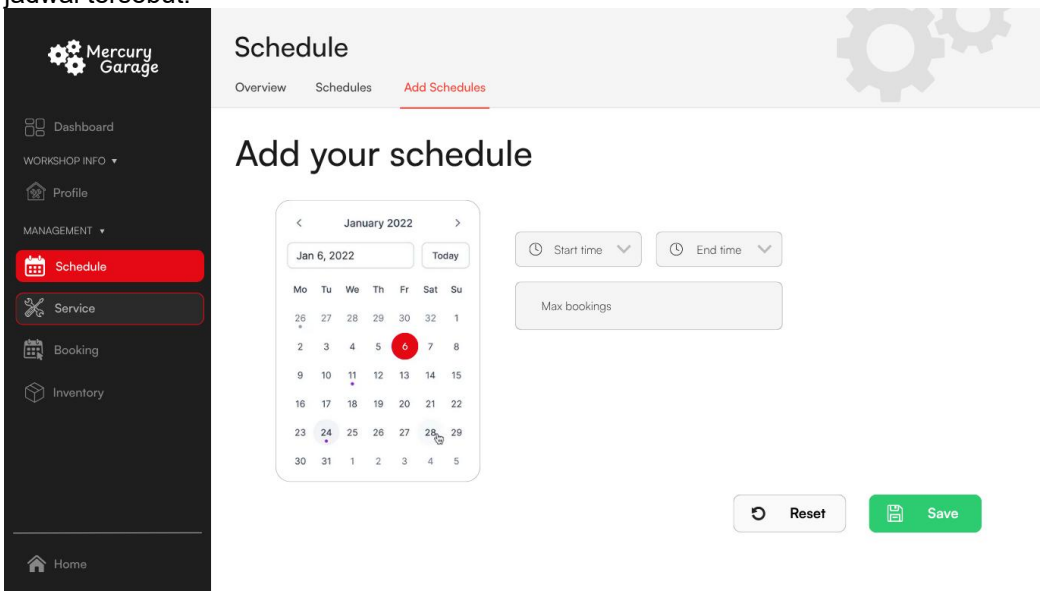
Halaman Manajemen Jadwal berfungsi untuk menampilkan dan mengelola daftar jadwal yang dibuat oleh pemilik bengkel. Pada halaman ini, pemilik bengkel dapat melihat informasi jadwal secara rinci seperti tanggal jadwal, jam mulai dan jam selesai jadwal. maksimal booking dan booking saat ini.



Gambar 18 Halaman manajemen jadwal

2.8.12 Halaman Tambah Jadwal

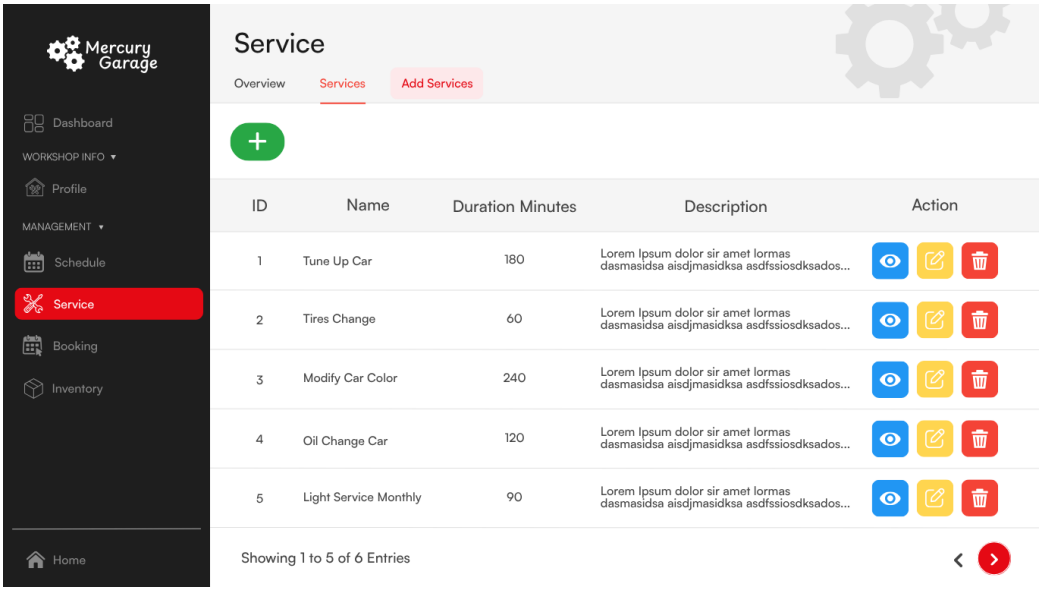
Halaman Tambah Jadwal (*Add Schedule*) digunakan oleh pemilik bengkel untuk menambahkan jadwal baru ke dalam sistem. Pada halaman ini, pemilik bengkel dapat memilih tanggal, menentukan jam mulai dan jam selesai, dan maksimal booking pada jadwal tersebut.



Gambar 19 Halaman tambah jadwal

2.8.13 Halaman Manajemen Layanan

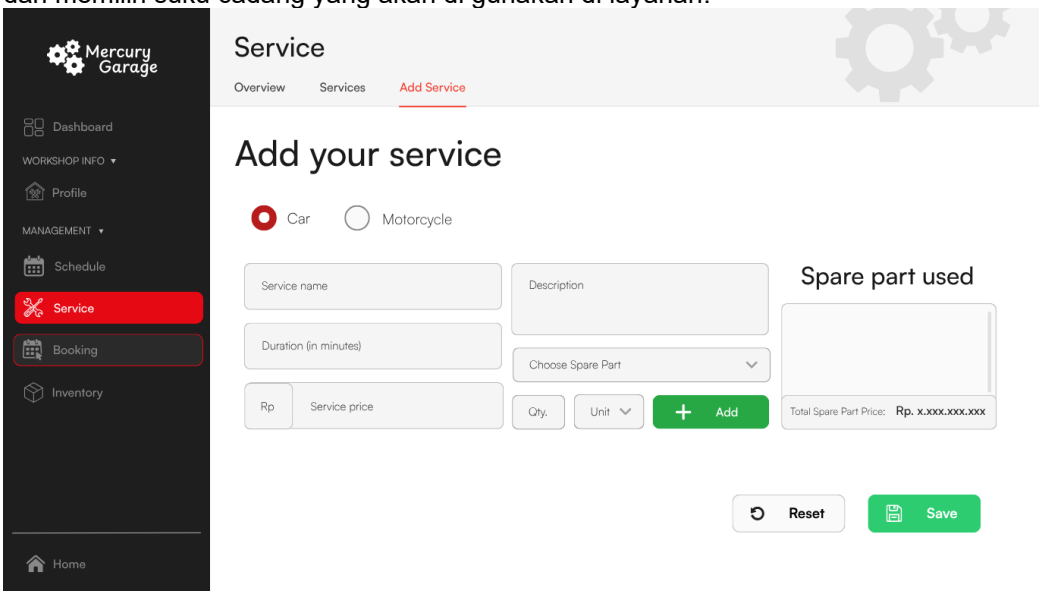
Halaman Manajemen Layanan berfungsi untuk menampilkan dan mengelola daftar layanan yang dibuat oleh pemilik bengkel. Pada halaman ini, pemilik bengkel dapat melihat informasi layanan secara rinci seperti nama layanan, durasi menit layanan, dan deskripsi layanan.



Gambar 20 Halaman manajemen layanan

2.8.14 Halaman Tambah Layanan

Halaman Tambah Layanan (*Add Service*) digunakan oleh pemilik bengkel untuk menambahkan layanan baru ke dalam sistem. Pada halaman ini, pemilik bengkel dapat memasukkan nama layanan, deskripsi layanan, durasi menit layanan, harga layanan, dan memilih suku cadang yang akan di gunakan di layanan.

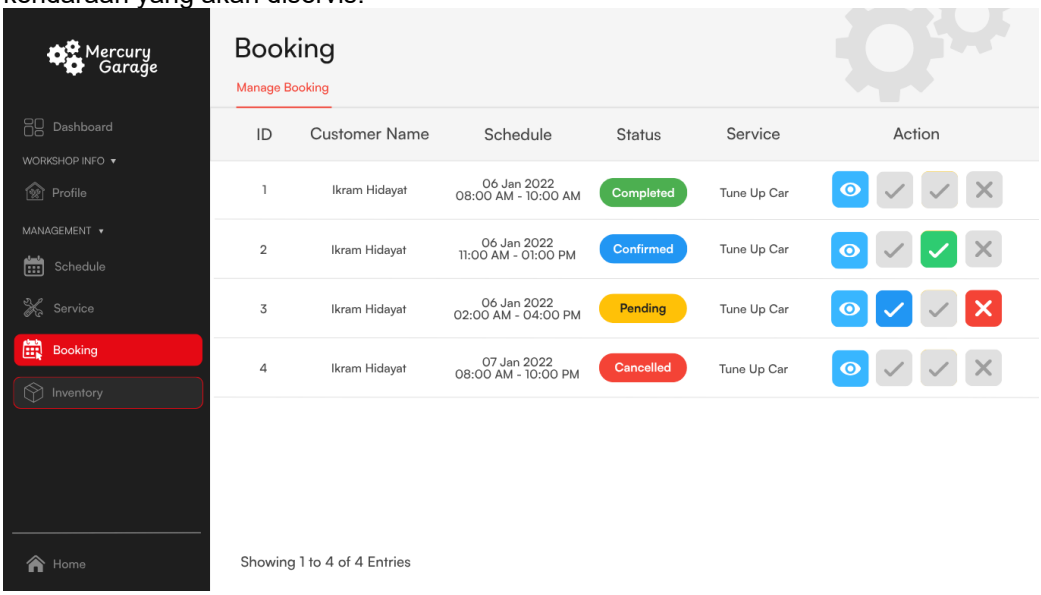


Gambar 21 Halaman tambah layanan

2.8.15 Halaman Manajemen *Booking*

Halaman Manajemen *Booking* berfungsi untuk menampilkan dan mengelola daftar *booking* yang dilakukan oleh pengguna yang memesan layanan di bengkel. Pada halaman ini, pemilik bengkel dapat melihat informasi *booking* secara rinci seperti ID dan

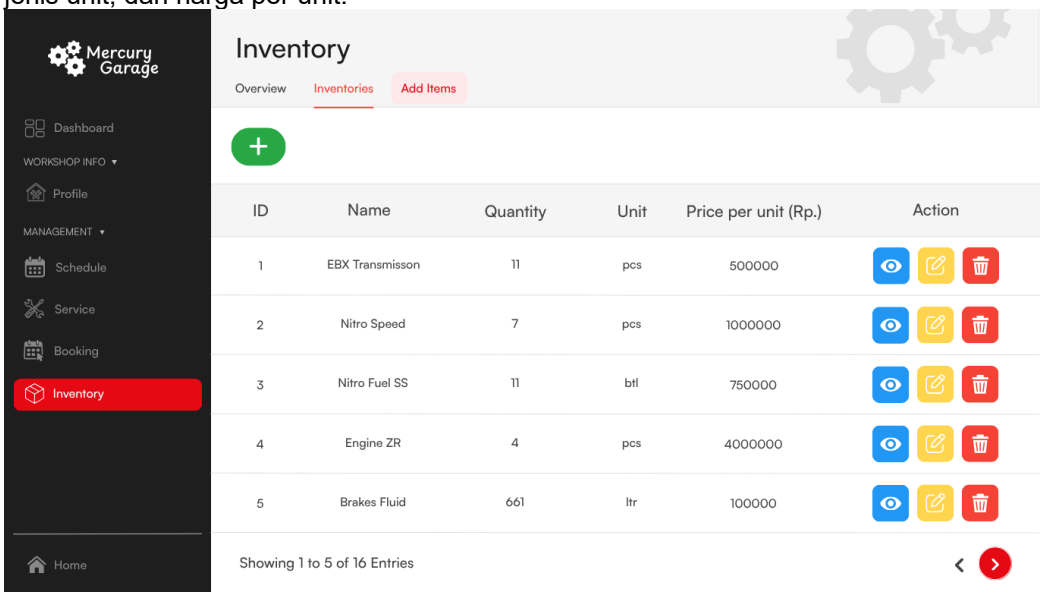
nama pelanggan, jadwal *booking*, status *booking*, layanan yang dipesan, serta kendaraan yang akan diservis.



Gambar 22 Halaman manajemen *booking*

2.8.16 Halaman Manajemen Inventaris

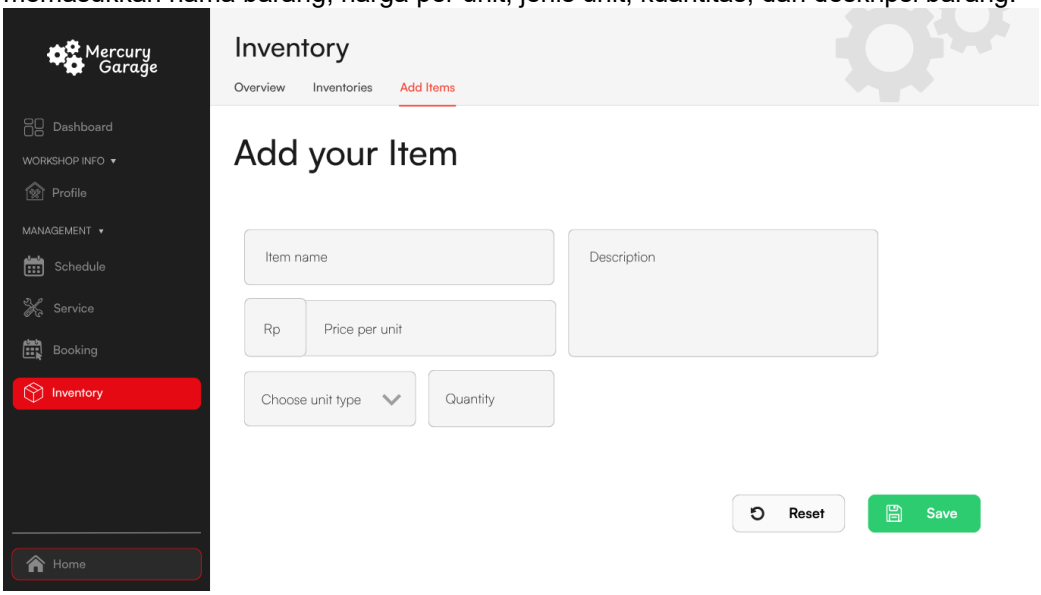
Halaman Manajemen Inventaris berfungsi untuk menampilkan dan mengelola daftar inventaris yang ditambahkan oleh pemilik bengkel. Pada halaman ini, pemilik bengkel dapat melihat informasi inventaris secara rinci seperti id dan nama barang, kuantitas, jenis unit, dan harga per unit.



Gambar 23 Halaman manajemen inventaris

2.8.17 Halaman Tambah Inventaris

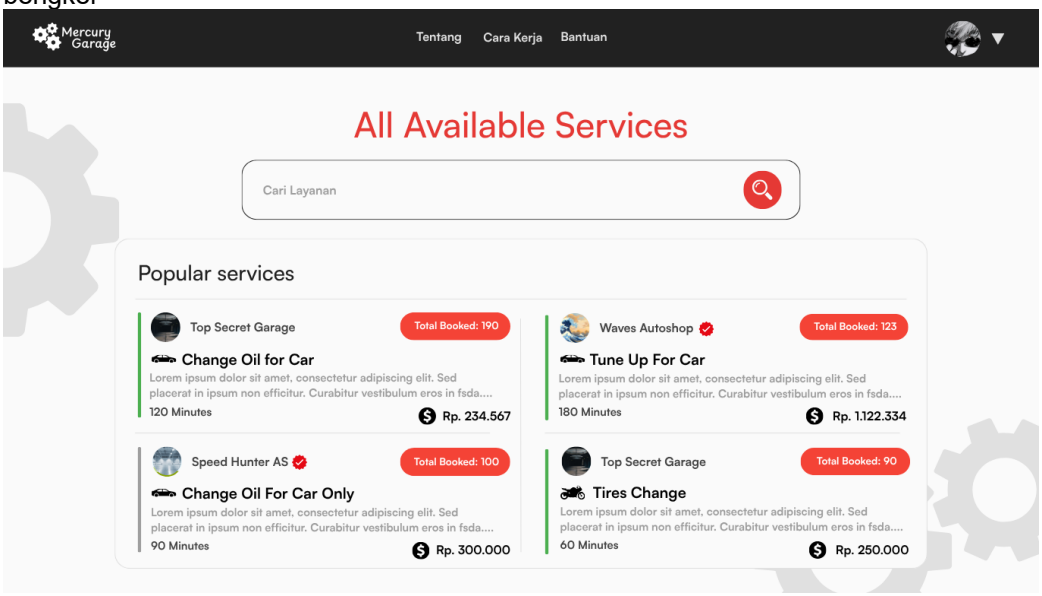
Halaman Tambah Inventaris (*Add Item*) digunakan oleh pemilik bengkel untuk menambahkan barang baru ke dalam sistem. Pada halaman ini, pemilik bengkel dapat memasukkan nama barang, harga per unit, jenis unit, kuantitas, dan deskripsi barang.



Gambar 24 Halaman tambah inventaris

2.8.18 Halaman Daftar Layanan

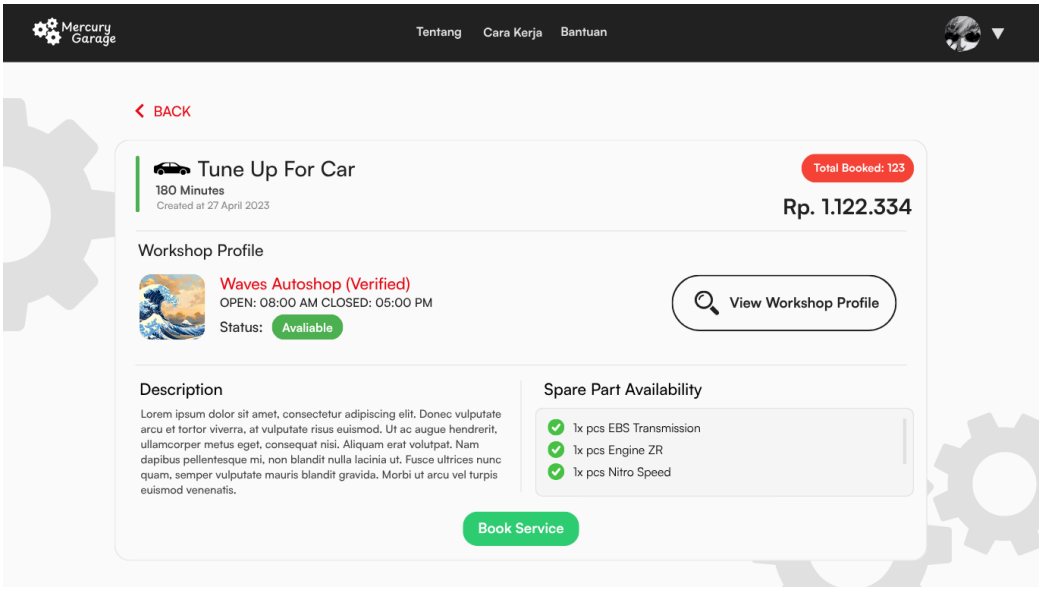
Halaman Daftar Layanan menampilkan semua layanan yang telah ditambahkan oleh bengkel



Gambar 25 Halaman daftar layanan

2.8.19 Halaman Detail Layanan

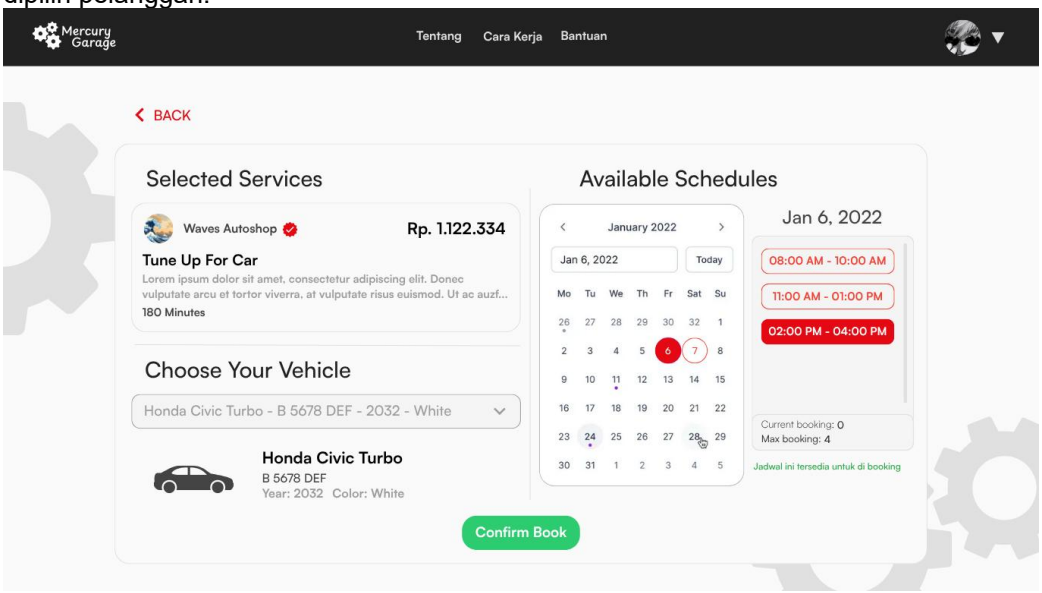
Halaman Detail Layanan menampilkan detail dari suatu layanan meliputi nama layanan, durasi menit layanan, harga layanan, profil bengkel yang membuat layanan, deskripsi layanan, dan ketersediaan suku cadang yang akan digunakan per 1 layanan.



Gambar 26 Halaman detail layanan

2.8.20 Halaman Detail *Booking*

Halaman Detail *Booking* menampilkan layanan yang dipilih untuk di *booking*, kendaraan yang dapat dipilih untuk di servis, dan jadwal *booking* yang telah dibuat bengkel untuk dipilih pelanggan.



Gambar 27 Halaman detail *booking*