Design of a Web-Based Chery Car Service Booking System at Arta Kranji Dealership, Bekasi

Jefry Arya Pratama[1], Muhammad Muharrom[2], Rahmiliasari Samnufida[3]
Information Technology Study Program, Universitas Bina Sarana Informatika, Indonesia

Vehicle service booking requires integrated management of customer, vehicle, service, schedule, and work-status data so that customers can obtain clear service information while workshop staff can manage operational capacity in a structured manner. This study aims to design and implement a web-based car service booking system that integrates three user roles: Customer, Service Advisor, and Super Admin. Data were collected through research activities at Chery Arta Kranji Dealer and requirement validation through observation and interviews with administrative staff. The Waterfall model was used for requirements analysis, design, implementation, testing, and support. The system was developed using PHP 8.2, Laravel 12, Inertia.js 2, React 19, TypeScript, Tailwind CSS 4, and MySQL 8. The implementation includes vehicle registration, a four-step booking process, a minimum H-1 booking rule and a maximum 60-day booking period, a two-booking quota per time slot, controlled status transitions, rescheduling with history records, booking tracking, invoices, and click-to-chat WhatsApp communication. Functional testing using black-box testing on 24 scenarios showed that all tested functions operated according to the expected results. The integration of three user roles and system-controlled booking rules establishes a structured service workflow and maintains slot-capacity consistency.
Keywords: Black-Box Testing, Information System, Service Booking, Waterfall, Web Application

This is an open access article under theCC BY-NClicense

Corresponding Author:
Jefry Arya Pratama
Information Technology Study Program, Universitas Bina Sarana Informatika, Indonesia

1. Introduction

The digitalization of automotive after-sales services has transformed the way workshops manage customer interactions. The service process involves more than simply bringing a vehicle to a workshop; it also encompasses customer identities, vehicle data, service package selections, time-slot availability, technician capacity, changes in service status, service histories, and post-booking communication. When this information is dispersed across WhatsApp conversations, separate records, or manual confirmations, customers may find it difficult to consistently determine the status of their service requests, while staff must consolidate operational data before making decisions. A web-based service booking system represents one approach to integrating this information into a single, traceable workflow.

Aisyah et al. [1] developed a web-based service booking information system for an automotive company to support booking processes and service data management [2] applied the Rapid Application Development method to a vehicle service booking system, while Diantara et al. [3] employed the Software Development Life Cycle approach to develop an online service booking application. Pangalila and Ardhiansyah [4] also demonstrated that web-based workshop booking management can address issues related to unclear queues, mechanic information, and service lists. These findings indicate that reservation digitalization has become a recurring necessity in workshop settings; however, implementation approaches may vary according to the business process characteristics of each research object.

In the context of car servicing, Supriatna and Tuslaela [5] designed a web-based vehicle service reservation system to reduce manual recordkeeping and queues at workshops. Setiawan et al.[6] developed a car

consultation and service booking system involving customer, admin, and technician roles using Laravel and MySQL. Artanti et al. [7] emphasized object-oriented design through use case, activity, sequence, class, component, and deployment diagrams. Farizqi et al.[8] and Pangestu and Fachrie [9] extended the application of reservation systems to mobile platforms. Maulana et al. [10] demonstrated that service booking applications can also support greater flexibility in workshop scheduling, while Ridwan [11] positioned online reservation services as a means of reducing the need for customers to visit workshops in person solely to obtain service appointments.

More recent developments indicate that workshop systems are evolving from simple reservation functions toward integrated service management. Arianti [12] developed a service booking application to facilitate appointment scheduling. Fera et al. [13] employed the Waterfall method to develop a workshop booking and scheduling website and demonstrated that the system could coordinate service schedules between customers and mechanics. Laia et al. [14] integrated service booking with real-time service progress monitoring, customer and vehicle data, payments, service histories, and reports. Daelami et al. [15] introduced daily service slot restrictions to reduce the risk of overbooking. Rifqi et al. [16] developed a multiplatform reservation system integrating workshop services, transactions, and tracking. Setiawan et al. [17] emphasized the integration of service administration, vehicle histories, transactions, and inventory within a web-based workshop management system.

These studies demonstrate that research on service booking has encompassed various platforms and development methods. Therefore, this study does not claim novelty merely on the basis of implementing a booking function. Instead, it focuses on integrating the service workflow through three user roles, namely Customer, Service Advisor, and Super Admin, as well as operational rules centrally controlled by the system. The Customer manages accounts, vehicles, bookings, service histories, status tracking, and invoices. The Service Advisor focuses on daily operations, including slot occupancy, walk-in bookings, status changes, WhatsApp follow-ups, and invoices. The Super Admin has additional authority to manage master data, internal users, catalogs, content, reports, and activity logs. This division is designed to establish clear boundaries of responsibility among customer-facing, operational, and advanced administrative functions.

The research object is Chery Arta Kranji Dealer in Bekasi. The dealer is listed on Chery's official dealer channel as a location providing sales, service, and spare parts [24]. News coverage of the dealer's opening also describes its 3S service concept and management by Arta Group [25]. This publicly available information is used solely to establish the research context. The findings concerning system requirements are derived from research activities, observations, and interviews with administrative staff; therefore, claims regarding operational problems are not based on the company's promotional materials.

Requirements validation indicated the need for a platform that enables customers to select vehicles, service packages, service dates, and appointment times, while helping staff verify scheduling capacity and follow up on bookings. This study does not use complaint frequency figures, problem percentages, average queue durations, or time-saving estimates because such quantitative data were unavailable. Accordingly, the evaluation focuses on the extent to which the system's functionality meets the identified requirements rather than measuring changes in organizational productivity before and after implementation.

Based on these issues, this study aims to design and implement a web-based car service booking system that integrates three user roles and centrally enforces booking rules. These rules cover booking limits, operating days, slot quotas, status changes, rescheduling, cancellation, tracking, invoicing, and operational communication through WhatsApp click-to-chat. In addition to implementing the application, the study conducts black-box testing to assess functional compliance with positive and negative test scenarios derived from the identified requirements, validation procedures, access policies, and business rules.

Tabel 1. Ringkasan Penelitian Terkait

Study	Year	Platform/Method	Focus	Position in Relation to the Present Study
Aisyah et al. [1]	2022	Web	Vehicle service booking	Foundation for the digitalization of the booking process
Dani & Eriana [2]	2022	Web / RAD	Vehicle booking	Comparison of development methods
Pangalila & Ardhiansyah [4]	2022	Web / Prototype	Workshop booking management	Comparison of service management approaches
Supriatna & Tuslaela [5]	2023	Web	Car service reservations	Closely related to the car service context
Setiawan et al. [6]	2024	Web / Waterfall	Car consultation and booking	Incorporates multiple user groups
Artanti et al. [7]	2024	Web / UML	Service booking design	Comparison of object-oriented modeling
Fera et al. [13]	2026	Web / Waterfall	Booking scheduling	Closely related to scheduling arrangements
Daelami et al. [15]	2026	Web / Waterfall	Booking and slot restrictions	Closely related to capacity control
This Study	2026	Web / Waterfall	Three user roles and centralized booking rules	Integration of customer interactions and operational processes

2. Methods

This study employs the Waterfall software development model because requirements, design, implementation, and testing are documented sequentially. In this study, Waterfall serves as a development framework rather than a basis for claiming that other methods are unsuitable. The main stages consist of requirements analysis, design, implementation, testing, and support. Each stage produces artifacts used in the subsequent stage; however, requirements are clarified whenever discrepancies between documentation and implementation are identified.

The official research period at the research site was recorded as running from September 2, 2025, to December 2, 2025. Following this period, additional validation was conducted on May 11, 2026, through observations of the service context and focused interviews with administrative staff. This distinction was made to prevent the official research period from being conflated with subsequent requirements validation activities. Primary data were used to understand the context of the service booking process, while project documentation and source code were used to describe the technical functions actually implemented in the application.

Data collection techniques consisted of observation, interviews, documentation studies, and literature reviews. Observation was conducted to understand the context of after-sales services and the administrative workflow for service bookings. Interviews were used to validate key requirements related to customers, vehicles, service package options, schedules, and staff follow-up activities. Documentation studies examined the academic outline, use case scenarios, diagram designs, interface screenshots, data specifications, and program implementation. The literature review positioned this study in relation to previous research on vehicle reservations, system development, and software testing.

Requirements analysis was conducted by distinguishing between functional and non-functional requirements. Functional requirements were categorized according to three user roles. Customers require registration, authentication, profile management, vehicle management, a four-step booking process, booking details and timelines, rescheduling, cancellation, invoices, and public tracking. Service Advisors require an operational dashboard, schedule and slot occupancy information, booking searches, walk-in booking creation, status updates, WhatsApp follow-up, and invoice management. Super Admins have the same operational functions, along with additional authority over master data, content, internal users, revenue reports, and activity logs.

Non-functional requirements include browser-based application access, role-based access control, server-side validation, time consistency within the Asia/Jakarta time zone, logging of specific activities, and the use of a relational database. Access control is enforced not only by hiding menus in the user interface but also through middleware and application-side policies, enabling the system to reject direct requests to unauthorized resources.

System design employs the Unified Modeling Language (UML) to describe application interactions and structure. The use case diagram illustrates the relationships between the three actors and the available functions. Activity diagrams describe activity flows, while sequence diagrams illustrate the order of interactions among actors, interfaces, controllers, services, models, and the database. Class diagrams, Entity Relationship Diagrams, and Logical Record Structures are used to explain data structures and relationships. This approach is consistent with the use of UML in the study by Artanti et al [7], however, the artifacts in the present study are tailored to the Laravel implementation and the service booking domain under investigation.

The implementation uses PHP 8.2 and Laravel 12 on the server side, Inertia.js 2 to connect the server application with the React interface, React 19 and TypeScript for the user interface, Tailwind CSS 4 for styling, and MySQL 8 as the database. Media files are stored using an external service, while WhatsApp communication uses a click-to-chat mechanism through wa.me links. This mechanism is not treated as the WhatsApp API because the server does not send messages automatically; instead, the system prepares the destination and message draft, which are subsequently opened by the user.

Testing focuses on black-box testing. This approach was selected because the study aims to examine the relationship between input conditions and functional outputs without treating the internal code structure as the object of evaluation. Fahrezi et al.[18] employed black-box testing to examine the functions of a web-based application, while Wintana et al. [19] discussed comparisons between testing approaches in terms of effectiveness. Triady et al. [20] applied equivalence partitioning to a website and demonstrated that test scenarios derived from input domains can identify functional discrepancies. Abdillah et al. [21] and Maulana and Voutama [22] also positioned black-box testing as a method for evaluating website functionality from an input-output behavior perspective. Huda et al. [23] used equivalence partitions in a cashier application to examine whether functions responded to test data in accordance with specifications.

The test scenarios in this study were derived from request validation, policies, booking configurations, status rules, and the service layer. Testing covered positive and negative cases, including valid data registration, duplicate email addresses, login attempts using active and inactive accounts, vehicle validation, booking date limits, closed days, fully occupied slots, concurrent requests for the last available slot, walk-in bookings, status transitions, cancellations, rescheduling, access control, the invoice lifecycle, WhatsApp draft generation, and public tracking. A test result was considered PASS when the actual output matched the expected result.

Development Stages

The Waterfall stages in this study can be summarized as follows: **Requirements Analysis** → **Design** → **Implementation** → **Testing** → **Support**. During the requirements analysis stage, field data and operational rules are translated into actor requirements and business rules. The design stage produces system models, database schemas, and interface designs. The implementation stage translates the design into a Laravel-Inertia-React application. The testing stage evaluates functionality through 24 black-box testing scenarios. The support stage covers application environment configuration and deployment documentation. This sequence ensures traceability among requirements, design, implementation, and testing.

Access Control Design

Table 2. Summary of Access Rights for the Three Roles

Role	Main Functions
Customer	Registration/login; profile management; vehicle management; booking; booking history; rescheduling/cancellation; tracking; invoices
Service Advisor	Operational dashboard; scheduling and slot occupancy; walk-in bookings; booking details; status updates; WhatsApp; invoices; reports
Super Admin	All authorized operational functions, as well as master data, catalog, content, internal users, revenue reports, invoice cancellation, and activity logs

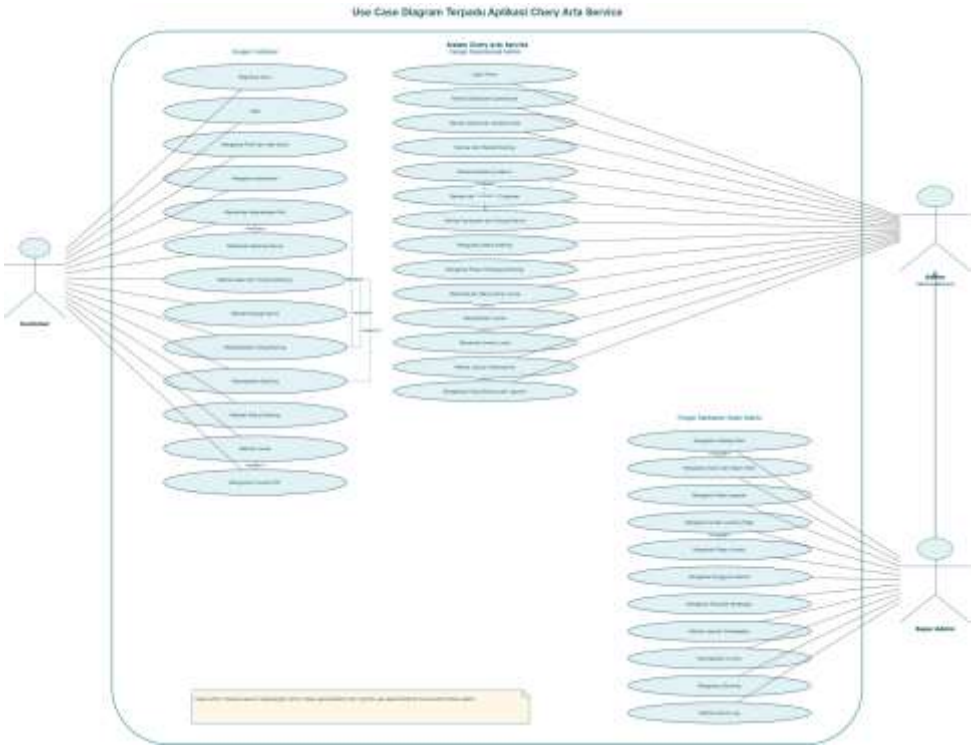

Figure 1. Use Case Diagram of the Three User Roles
Source: Research Findings (2026)

Figure 1 illustrates that Customer, Service Advisor, and Super Admin have distinct groups of functions. The diagram is used to maintain consistency between requirements and implementation, particularly for functions related to customer data, status changes, invoices, and advanced administration. Super Admin is not treated as an actor identical to Service Advisor, but rather as a role with additional authority over system management.

Booking Business Rules

Business rules are positioned as a core component of the design because schedule availability cannot be determined solely by the calendar interface. Dates and times follow the Asia/Jakarta time zone. Web bookings require a minimum lead time of one day in advance (H-1), whereas walk-in bookings created by Service Advisors can be made on the same day. Dates more than 60 days in the future are rejected. No booking slots are available on Sundays. The standard capacity is two bookings per slot, while the *cancelled* and *no_show* statuses release the reserved capacity. From Monday to Friday, slots are available during the designated operating hours at 30-minute intervals, whereas Saturday operating hours end earlier. Self-service rescheduling and cancellation are permitted only for eligible booking statuses and must be performed at least one day in advance (H-1).

Table 3. Main Booking Rules

Rule	Value	Notes
Time zone	Asia/Jakarta	All dates and times follow the workshop's local time
Web booking lead time	Minimum H-1	Walk-in bookings by Service Advisors may be made on the same day
Advance booking limit	60 days	Dates beyond this limit are rejected by the server
Closing day	Sunday	No booking slots are offered
Capacity	2 bookings per slot	*Cancelled* and *no_show* bookings do not count toward capacity
Monday–Friday	08:00–11:30; 13:00–14:00	30-minute intervals; 12:00 and 12:30 are unavailable
Saturday	08:00–11:30; 13:00	Adjusted to the workshop's operating hours
Rescheduling/cancellation	*Pending*/*confirmed* status and at least H-1	A cancellation reason is mandatory

3. Result And Discussion

The research resulted in a web-based service booking application that integrates customer data, vehicles, vehicle models, service packages, schedules, statuses, booking history, communication, and invoices. The system not only provides a public-facing interface for customers but also an operational panel for Service Advisors and an advanced administrative panel for Super Admins. The implementation is built using a modern monolithic architecture based on Laravel and Inertia, allowing routing, controllers, services, models, policies, and the database to be managed within a single application, while React is used to provide an interactive user interface.

At the user layer, Customers access the system through a browser on a smartphone or laptop. Service Advisors and Super Admins also use browsers on internal devices. Requests are transmitted via HTTPS to the application service. The MySQL database stores user, vehicle, booking, invoice, and content data. Media files are stored separately through an external service. WhatsApp communication is directed through wa.me links, allowing users to initiate message sending manually. This design distinguishes the application's internal dependencies from external services and avoids claiming that the system uses automated WhatsApp API integration.

This architecture supports the separation of responsibilities among routes, controllers, services, models, and policies. Request validation handles the structure of input data, the service layer manages business rules such as slot capacity and status transitions, models handle data relationships, and policies verify

authorization for specific resources. This approach is important in a three-role system because restricting menus in the user interface alone is insufficient to prevent unauthorized access.

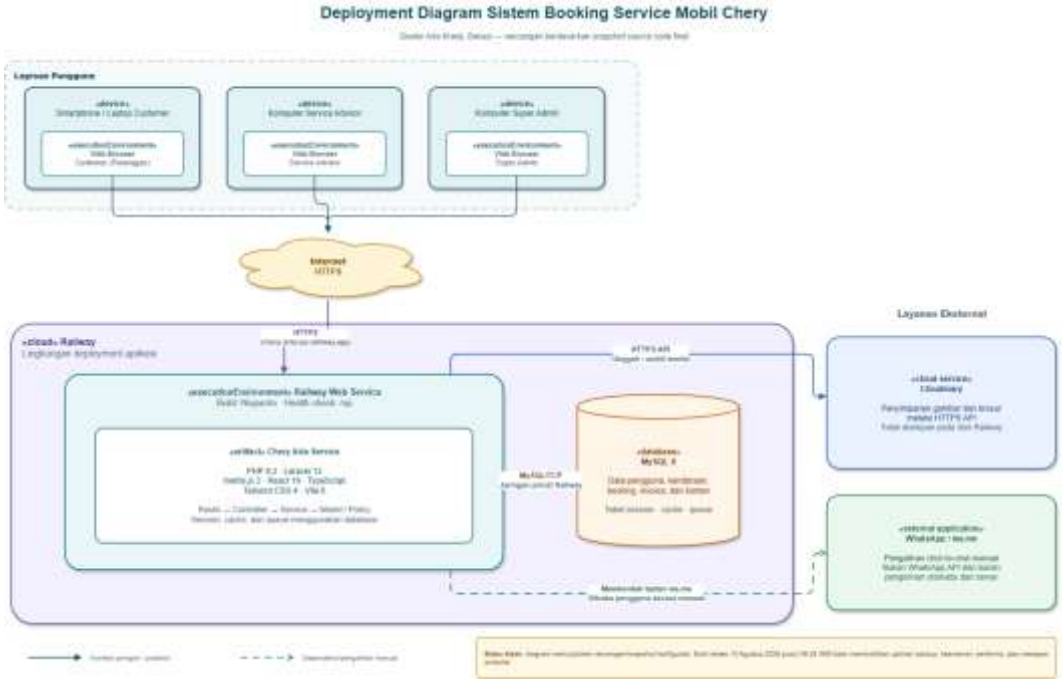

Figure 2. Deployment Diagram of the Service Booking System
Source: Research Findings (2026*)*

Figure 2 illustrates the application's deployment within a web environment. The diagram serves as an overview of the implementation configuration and is not used to draw conclusions about uptime, performance, security, backups, or production readiness, none of which were specifically tested. This limitation is important to ensure that the research findings remain consistent with the available evidence.

Implementation of the Customer Booking Process

The Customer booking process is organized into four stages to avoid concentrating all data entry on a single page. The first stage involves selecting a vehicle registered to the account. The second stage involves selecting an active service package. The third stage involves selecting a date and time based on the calendar and available slots. The fourth stage presents a summary to ensure that the vehicle, service, schedule, odometer reading, and complaint details are correct before the request is saved.

When a Customer submits a booking, the system verifies vehicle ownership, service package status, date, time, and slot capacity. Capacity verification is performed on the server side. The implementation also uses database transactions and lockForUpdate() when calculating capacity, ensuring that two requests submitted almost simultaneously do not rely solely on the slot status previously displayed in the browser. If capacity has been reached, the system rejects the booking and may direct the user to select an alternative time. If the request is valid, the system creates a booking with a unique code and records its initial status in the booking history.

The implementation of server-side capacity limits extends the slot restriction functionality also discussed by Daelami et al. [15]. In this study, capacity is determined not only by the number of booking records but also by the statuses that occupy capacity. Bookings with *cancelled* and *no_show* statuses are not considered to occupy a slot. This logic ensures that schedule availability remains consistent with the booking lifecycle rather than being based solely on the total number of bookings previously created for a particular time.

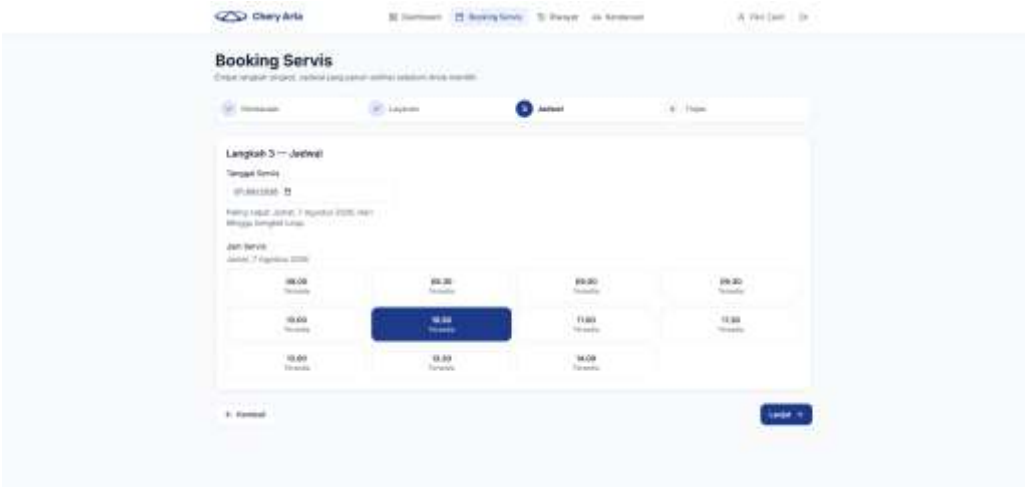

Figure 3. Booking Schedule Selection Interface
Source: Research Findings (2026)

Figure 3 illustrates the schedule selection stage. The interface displays available time slots based on the selected date; however, the server makes the final determination regarding validity. Separating the information displayed in the interface from server-side validation reduces dependence on browser conditions that may change or rely on outdated data.

Implementation of Service Advisor and Super Admin Functions

Service Advisors use the operational dashboard to view booking summaries, follow-up priorities, daily schedules, and service statuses. The scheduling page displays slot occupancy, allowing staff to identify the capacity already used on a particular date. In addition to bookings created by Customers, Service Advisors can create walk-in bookings for customers who arrive directly at the workshop. The main difference between web bookings and walk-in bookings concerns the lead time: web bookings require a minimum lead time of one day (H-1), whereas walk-in bookings can be created for the same day, provided that slots are available.

On the booking details page, Service Advisors can view vehicle information and service requirements, record operational information, and update statuses in accordance with valid transitions. The system does not permit arbitrary status transitions. For example, a *pending* booking may be confirmed or cancelled, but a direct transition from *pending* to *completed* is rejected. When certain conditions are met, information such as confirmed_at, the staff member handling the booking, and the change history is also recorded. State transitions help ensure that each status accurately represents a process that has actually occurred.

Super Admins have additional authority to manage data not required for the daily work of Service Advisors, such as the car catalog, service packages, internal users, message templates, public content, specific reports, and activity logs. This separation is also examined during testing through scenarios in which Service Advisors attempt to access Super Admin menus. Such access is expected to be denied by the middleware or policy. Thus, roles function as access controls rather than merely as variations in menu presentation.

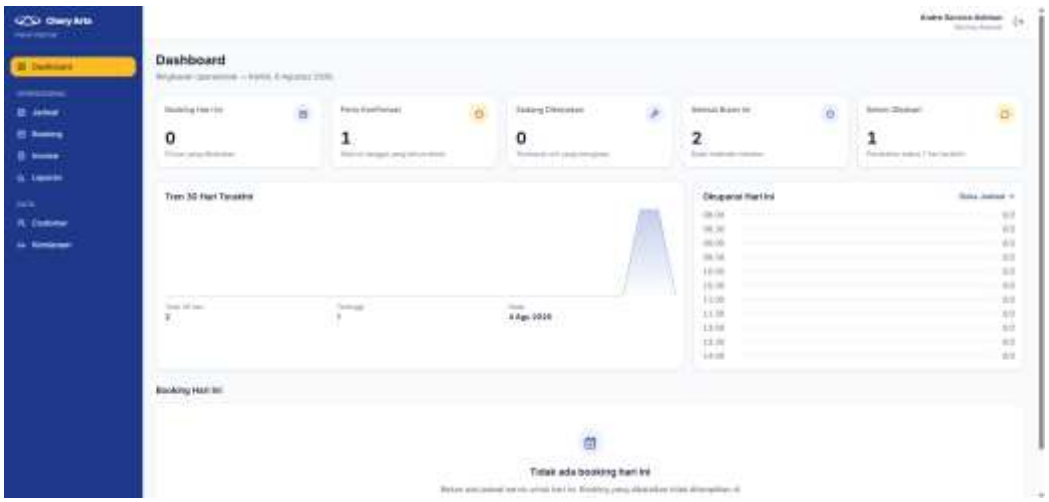

Figure 4. Service Advisor Dashboard
Source: Research Findings (2026)

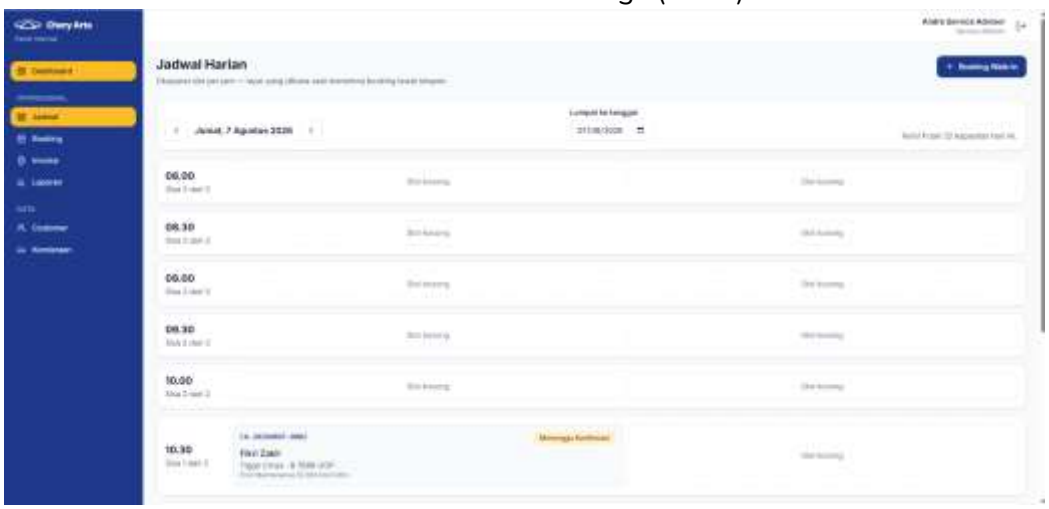

Figure 5. Schedule and Slot Occupancy
Source: Research Findings (2026)

Rescheduling, Tracking, Invoices, and WhatsApp

The rescheduling function is used when Customers need to change a booking that still meets the eligibility requirements. The system only permits rescheduling for bookings with a *pending* or *confirmed* status that also satisfy the H-1 requirement. The process does not overwrite all existing data. When the new slot is available, the system records the schedule change while retaining the previous booking record as part of the historical audit trail. If the destination slot is full, the transaction is rolled back and the original booking remains active. This approach maintains consistency if the change process fails during the transaction.

The public tracking function allows users to enter a booking code without logging in. The displayed information is restricted to a summary considered safe for public access. If the code is not found, the system does not display personal data. This restriction is important because the tracking page is accessible without authentication. Using a unique code as a reference helps customers check their booking status while avoiding the need to access broader internal data.

The invoice lifecycle is linked to the booking. When the process reaches the specified conditions, the system can provide an active, non-void invoice. The *void invoice* function requires a reason, ensuring that the cancellation of a transaction document is recorded and traceable. Meanwhile, WhatsApp follow-up uses active templates and normalized customer phone numbers. The system generates a wa.me link and records

a log when a message draft is created through the corresponding button. Messages are still sent manually through the WhatsApp application; therefore, this implementation is not equivalent to automated message delivery from the server.

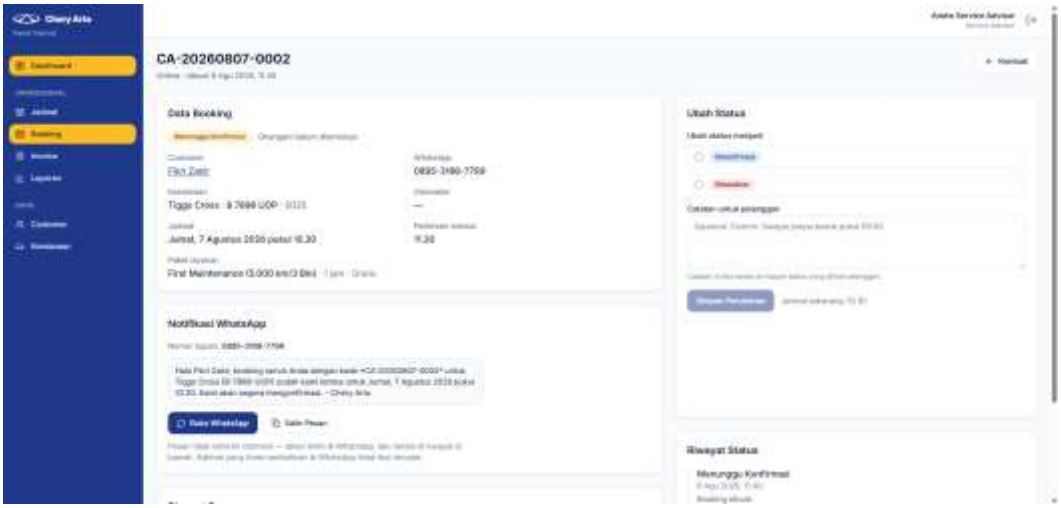

Figure 6. Booking Details and WhatsApp Click-to-Chat
Source: Research Findings (2026)

Black-Box Testing Results

Actual testing was conducted on 24 scenarios derived from the system requirements and rules. All scenarios produced actual outputs that matched the expected results and were therefore assigned a PASS status. Since documentation of execution dates and testing screenshots was unavailable, the study does not report dates or visual evidence that cannot be verified. The interpretation of the results is limited to functional conformance and is not used to draw conclusions about usability quality, performance under high loads, overall security, or user satisfaction levels.

Table 4. Black-Box Testing Results

ID	Scenario	Actual Result	Status
BT-01	Registration with unique email and WhatsApp number	Customer account created and session established	PASS
BT-02	Registration using an email address that is already in use	Request rejected and validation message displayed	PASS
BT-03	Login with valid credentials	Session regenerated and user redirected	PASS
BT-04	Login using an inactive account	Access denied	PASS
BT-05	Add a vehicle with a valid license plate	Vehicle saved and plate_full generated	PASS
BT-06	Add the same license plate for the same owner	Saving rejected	PASS
BT-07	Web booking made for H-1	Booking created with *pending* status and a unique code	PASS
BT-08	Web booking made for the same day	Date rejected and earliest available date information displayed	PASS
BT-09	Booking on a Sunday	Date rejected because the workshop is closed	PASS

BT-10	Booking more than 60 days in advance	Date rejected	PASS
BT-11	Booking a fully occupied slot	Saving rejected and alternative times offered	PASS
BT-12	Two simultaneous requests for the last available slot	Capacity remains limited to two bookings	PASS
BT-13	Same-day walk-in booking	Booking immediately confirmed	PASS
BT-14	Transition from *pending* to *confirmed*	Status, confirmation time, assigned staff member, and history recorded	PASS
BT-15	Direct transition from *pending* to *completed*	Request rejected	PASS
BT-16	Cancellation without a reason	Request rejected	PASS
BT-17	Rescheduling to an available slot	New booking created; previous booking marked *cancelled*; relationship recorded	PASS
BT-18	Rescheduling to a fully occupied slot	Transaction rolled back and previous booking remained active	PASS
BT-19	Customer attempts to access another customer's booking	Access denied by policy	PASS
BT-20	Service Advisor attempts to access the Super Admin menu	Access denied by middleware/policy	PASS
BT-21	Completion triggers creation of a draft invoice	One active, non-void invoice becomes available	PASS
BT-22	Voiding an invoice without a reason	Request rejected	PASS
BT-23	WhatsApp draft generated from an active template	wa.me link and generated log created	PASS
BT-24	Public tracking using a code that does not exist	No personal data displayed	PASS

The distribution of scenarios indicates that testing was not limited to the happy path. Negative test cases included duplicate email addresses, inactive accounts, invalid dates, closing days, full capacity, prohibited status transitions, cancellation without a reason, resource authorization, and tracking with a random code. Scenario BT-12 specifically examined two requests submitted for the last available slot to ensure that capacity did not exceed two bookings. The PASS results across all scenarios indicate that the behavior of the tested functions conformed to the established rules. However, these results do not constitute a guarantee that the system is free from errors under all possible operating conditions.

Discussion

The research findings indicate that the web-based *service booking* system can serve as a central platform for managing service workflows rather than merely functioning as a reservation form. Aisyah et al. [1], Supriatna and Tuslaela [5], and Pangalila and Ardhiansyah [4] all demonstrate the need to digitize booking processes, as manual procedures may lead to unclear queues and recording risks. The system developed in this study adopts the same general objective but extends its scope to vehicle management, slot rules, *booking* states, history, invoices, communication, and auditing. Consequently, the data used for operational decision-making originate from a unified workflow.

In terms of methodology, the use of the Waterfall model is consistent with the studies conducted by Setiawan et al [6], Fera et al. [13], Daelami et al. [15], and Rifqi et al. [16]. Waterfall was selected because the academic documentation structure requires traceability between requirements, design, implementation, and testing. Nevertheless, the technical implementation still requires validation against the actual *source code* to ensure that the documentation does not describe nonexistent functionality. Therefore, implementation decisions, such as capacity limits, *lead time*, and state transitions, are explained based on the established system behavior rather than solely on the initial design.

Slot management is essential because service schedules have limited capacity. Daelami et al. [15] implemented daily slot restrictions to reduce the risk of *overbooking*. This study applies a similar principle by limiting capacity to two *bookings* per slot while linking capacity calculations to booking status. Bookings marked as *cancelled* or *no_show* do not consume capacity. During *rescheduling*, the system must also release the original slot's capacity and verify the availability of the new slot within a consistent transaction. The relationship between capacity and the *booking* lifecycle makes the scheduling process more representative of actual conditions than simply counting the total number of bookings ever created.

The division into three user roles can be compared with the study by Setiawan et al. [6] which involved customers, administrators, and technicians. The difference is that this study positions the *Service Advisor* as the primary operational actor and the *Super Admin* as an advanced administrative actor. This separation is relevant to dealerships because daily service-handling activities do not always require the authority to manage internal users, catalogs, content, or *activity logs*. The results of BT-19 and BT-20 demonstrate that authorization testing is part of functional verification rather than merely an inspection of visible menus.

The *rescheduling* function and booking history provide traceability for schedule changes. A system that only updates the date in the same record may lose the context of a change when an operational audit is required. In this study, *rescheduling* preserves the relationship with the original *booking* and rolls back the transaction if the new slot is unavailable. This approach also supports capacity consistency because schedule changes are treated as a business process rather than a simple data edit.

The integration of invoices and WhatsApp supports processes that occur after a *booking* has been created. Laia et al.[14] demonstrate the tendency of modern workshop systems to integrate reservations with monitoring, payments, history, and reporting. Rifqi et al. [16] also combine reservations with transactions and tracking in a multiplatform system. This study adopts a more limited scope: invoices are managed within the system, while WhatsApp is used for manual *click-to-chat* communication. These limitations are explicitly stated to prevent the implementation from being misinterpreted as providing payment integration or automated messaging functionality that is not available.

From a modeling perspective, Artanti et al.[7] demonstrate that UML can be used to maintain consistency between analysis and design in a *service booking* system. In this study, UML is also used to represent actors, workflows, interaction sequences, and system structures. However, the article presents only the most relevant use case and deployment diagrams to ensure that the discussion is not dominated by the full set of design artifacts. Activity, sequence, class, ERD, and LRS diagrams remain part of the development documentation, but not all are necessary to explain the article's contributions.

Black-box testing was selected because the evaluation aims to examine user-visible functions and system rules. The studies by Fahrezi et al.[18], Triady et al. [20], Abdillah et al. [21], Maulana and Voutama [22], and Huda et al. [23] similarly demonstrate the use of *black-box* approaches in web applications or information systems. The present study differs in the domain being tested, namely *service booking* rules, user roles, and service transactions. The 24 scenarios provide coverage of core functions and negative conditions directly related to the research requirements.

All test scenarios received a PASS status. These results support the conclusion that the tested functions produce responses consistent with their specifications. However, PASS results in *black-box testing* cannot serve as evidence that the application is easy to use, delivers high performance, is secure against all classes of attacks, or increases workshop productivity by a specific percentage. Wintana et al. [19] emphasize that testing techniques have different characteristics and scopes. Accordingly, the claims of this study are limited to functional conformance and the behavior of business rules covered by the test scenarios.

Compared with the studies by Farizqi et al. [8], Pangestu and Fachrie [9], Maulana et al. [10], Ridwan [11], and Arianti [12] which demonstrate various web and mobile implementations, this study adopts a web-based platform so that customers and staff can access the system through a browser without requiring the distribution of a dedicated mobile application. This platform choice does not imply that web applications are superior under all conditions. The more significant contribution lies in how the system integrates user roles, capacity rules, status changes, and operational functions through a single data source.

In practical terms, this design provides a foundation for a more structured service workflow. Customers can access information about their vehicles and bookings, Service Advisors can obtain an operational overview of schedules and statuses, and Super Admins can manage system-level resources. Server-side controls for dates, slots, statuses, and policies reduce the likelihood of business rules being enforced only at the interface level. This is important because browsers can submit direct requests or use outdated data, whereas business decisions must remain consistent on the application side.

The main limitation of this study lies in the scope of its evaluation. Testing focused on functionality and did not include *usability testing*, User Acceptance Testing, load testing, penetration testing, or measurements of operational impact before and after implementation. In addition, visual evidence and actual test execution dates were unavailable for inclusion in the article; therefore, the test results are presented as a table of scenarios and statuses. These limitations do not invalidate the functional results but restrict the types of conclusions that can be drawn from the study.

4. Conclusion

This study developed a web-based car *service booking* system that integrates *Customer*, *Service Advisor*, and *Super Admin* roles within a unified service management workflow. The system connects customer data, vehicles, service packages, schedules, *booking* codes, statuses, history, invoices, operational communication, and administrative functions. Business rules are enforced at the system level through a minimum H-1 lead time for web bookings, a 60-day advance booking limit, Sunday as a closing day, a capacity of two bookings per slot, controlled status transitions, and *rescheduling* that preserves historical records. *Black-box* testing of 24 scenarios showed that all tested functions produced the expected outputs, including negative scenarios and access controls. These results support the conclusion that the system meets the established functional requirements and provides a more structured booking process. The main limitation of this study is that the evaluation remains focused on functionality. Future research could incorporate *usability testing*, User Acceptance Testing, performance testing, security testing, and measurements of operational impact under actual usage conditions to assess system quality from a broader perspective.

5. References

[1] E. S. Aisyah, R. D. Anjani, dan Y. N. Syahriani, “Perancangan Sistem Informasi Booking Service Berbasis Web Pada PT. Srikandi Diamond Motors,” Journal Sensi, vol. 8, no. 2, pp. 131–140, 2022, doi: 10.33050/sensi.v8i2.2413.

[2] D. Dani dan E. S. Eriana, "Perancangan Sistem Informasi Booking Service Kendaraan Dengan Metode Rapid Application Development," Sainstech: Jurnal Penelitian dan Pengkajian Sains dan Teknologi, vol. 32, no. 3, pp. 12–18, 2022, doi: 10.37277/stch.v32i3.1367.

[3] R. Diantara, Siswanto, dan Yupianti, "Perancangan Aplikasi Sistem Booking Service Online Berbasis Web dengan Menggunakan Metode Software Development Life Cycle," Jurnal Media Computer Science, vol. 1, no. 1, pp. 19–24, 2022.

[4] A. J. F. Pangalila dan M. Ardhiansyah, "Perancangan Sistem Aplikasi Management Booking Service Bengkel Berbasis Website Dengan Metode Prototype (Studi Kasus Bengkel Motor Rido Racing Rempoa Tangerang Selatan)," INFORMATIKA, vol. 3, no. 2, 2022.

[5] N. Supriatna dan Tuslaela, "Perancangan Sistem Informasi Reservasi Service Mobil Daihatsu dan Toyota Berbasis Website pada Liean Motors," Jurnal Ilmu Komputer dan Bisnis, vol. 14, no. 2, pp. 190–201, 2023, doi: 10.47927/jikb.v14i2.531.

[6] M. Setiawan, E. S. Pratama, E. I. Setiawan, dan T. P. Gunawan, "Sistem Konsultasi dan Booking Service Mobil Berbasis Web," Prosiding TAU SNARS-TEK Seminar Nasional Rekayasa dan Teknologi, vol. 2, no. 1, 2024, doi: 10.47970/snarstek.v2i1.603.

[7] K. D. Artanti, F. R. Bagastavi, A. D. C. Prasetya, dan A. S. Fitri, "Analisis Desain Berbasis Objek untuk Pengembangan Website Booking Service Motor Online Pada Bengkel Panji Motor Sidoarjo," Innovative: Journal of Social Science Research, vol. 4, no. 3, pp. 6083–6097, 2024, doi: 10.31004/innovative.v4i3.11084.

[8] M. R. Farizqi, A. D. Kalifia, dan R. H. P. Sejati, "Design of an Android-Based Motorcycle Service Booking Application," International Journal Software Engineering and Computer Science, vol. 4, no. 1, pp. 377–385, 2024, doi: 10.35870/ijsecs.v4i1.2334.

[9] R. A. Pangestu dan M. Fachrie, "Development of Motorcycle Workshop Reservation Application at Wahyu Motor, Godean," International Journal of Computer Applications, vol. 186, no. 2, pp. 1–7, 2024, doi: 10.5120/ijca2024923345.

[10] R. A. Maulana, M. Ancela, dan D. Vernanda, "Perancangan Sistem Informasi Booking Service Bengkel Motor Pata Service Motor Berbasis Aplikasi," Inventor: Jurnal Inovasi dan Tren Pendidikan Teknologi Informasi, vol. 2, no. 2, pp. 54–60, 2024, doi: 10.37630/inventor.v2i2.1331.

[11] A. N. S. Ridwan, "Desain Aplikasi Layanan Reservasi Bengkel Motor Online Pilihan Tepat untuk Servis Tanpa Repot," Uranus: Jurnal Ilmiah Teknik Elektro, Sains dan Informatika, vol. 2, no. 4, pp. 273–289, 2024, doi: 10.61132/uranus.v2i4.544.

[12] I. Arianti, "Penerapan Aplikasi Booking Jasa Service Berbasis Android Pada Bengkel Sehat Pro Auto Clinic," U-NET Jurnal Teknik Informatika, vol. 7, no. 2, pp. 1–11, 2025, doi: 10.52332/u-net.v7i2.566.

[13] F. Fera, D. David, dan F. A. Tarigan, "Perancangan Website Customer Service Penjadwalan Booking Pada Bengkel Acuan dengan Metode Waterfall," Jurnal Pendidikan Tambusai, vol. 10, no. 1, pp. 7434–7443, 2026, doi: 10.31004/jptam.v10i1.37469.

[14] A. Laia, D. I. Mulyana, dan M. B. Yel, "Digitalisasi UMKM Bengkel melalui Sistem Booking Servis dan Monitoring secara Real-Time," Jurnal Indonesia: Manajemen Informatika dan Komunikasi, vol. 7, no. 3, pp. 498–506, 2026, doi: 10.63447/jimik.v7i3.2014.

[15] F. M. Daelami, R. S. M. Maharani, dan A. A. Chamid, "Sistem Informasi Booking Layanan Pemasangan Lampu Variasi Berbasis Website pada Bengkel DVM Motor," Jurnal Sistem Informasi Triguna Dharma, vol. 5, no. 1, pp. 21–29, 2026, doi: 10.53513/jursi.v5i1.12368.

[16] M. Rifqi, F. Fadhillah, Syauki, Y. Afif, U. Najwa, dan A. Pratama, "Design of a Multi-Platform Motorcycle Workshop E-Commerce and Reservation System with Payment Gateway Integration," Journal of Artificial Intelligence and Engineering Applications, vol. 5, no. 3, pp. 4838–4844, 2026, doi: 10.59934/jaiea.v5i3.2548.

[17] M. D. Setiawan, G. G. Setiaji, dan A. Rifa'i, "Rancang Bangun Sistem Informasi Manajemen Service Kendaraan Berbasis Web Bengkel Menggunakan Metode Prototype," TIN: Terapan Informatika Nusantara, vol. 7, no. 1, pp. 196–207, 2026, doi: 10.47065/tin.v7i1.10059.

[18] A. Fahrezi, F. N. Salam, G. M. Ibrahim, R. R. Syaiful, dan A. Saifudin, "Pengujian Black Box Testing pada Aplikasi Inventori Barang Berbasis Web di PT. AINO Indonesia," LOGIC: Jurnal Ilmu Komputer dan Pendidikan, vol. 1, no. 1, pp. 1–5, 2022.

[19] D. Wintana, D. Pribadi, dan M. Y. Nurhadi, "Analisis Perbandingan Efektifitas White-Box Testing dan Black-Box Testing," Jurnal Larik: Ladang Artikel Ilmu Komputer, vol. 2, no. 1, pp. 8–16, 2022, doi: 10.31294/larik.v2i1.1382.

[20] D. Triady, I. A. Musdar, dan H. Surasa, "Blackbox Testing on Worker's Website Using Equivalence Partitioning Method," KHARISMA Tech, vol. 18, no. 1, pp. 84–98, 2023, doi: 10.55645/kharismatech.v18i1.292.

[21] M. T. Abdillah, I. Kurniastuti, F. A. Susanto, dan F. Yudianto, "Implementasi Black Box Testing dan Usability Testing pada Website Sekolah MI Miftahul Ulum Warugunung Surabaya," Jurnal Ilmu Komputer dan Desain Komunikasi Visual, vol. 8, no. 1, pp. 234–242, 2023, doi: 10.55732/jikdiskomvis.v8i1.897.

[22] T. Maulana dan A. Voutama, "Black Box Testing dengan Teknik Equivalence Partitions pada Website Dinas Pengendalian Penduduk dan Keluarga Berencana Karawang," Jurnal Sistem Informasi dan Informatika (Simika), vol. 6, no. 2, 2023, doi: 10.47080/simika.v6i2.2536.

[23] M. N. Huda, M. Burhan, A. Satibi, H. A. Pradita, A. Saifudin, dan I. Kusyadi, "Implementasi Black Box Testing pada Aplikasi Sistem Kasir dengan Menggunakan Teknik Equivalence Partitions," Jurnal Teknologi Sistem Informasi dan Aplikasi, vol. 5, no. 2, pp. 120–127, 2022, doi: 10.32493/jtsi.v5i2.17645.

[24] Chery Indonesia, "Dealer Chery," 2026. https://chery.co.id/id/dealer (diakses 2026).

[25] SWA, "Chery Tancap Gas, Buka Dealer Terbesar di Jabodetabek," 2025. https://swa.co.id/read/459586/chery-tancap-gas-buka-dealer-terbesar-di-jabodetabek (diakses 2026).