Faktor Exacta

Vol. 19, No. 1, March 2026, pp. 60~72
eISSN: 2502-339X, pISSN: 1979-276X, https://doi.org/10.30998/faktorexacta.v19i1.1001

Development of Web Application Functionality for the Driver Management System at Katadata

Muhammad Bagit[1] Dimas Tri Raharjo[2]
[1,2]Program Studi Sistem Informasi, Sekolah Tinggi Teknologi NIIT, Indonesia

Article Info

Article history:

Received 12/11/2025
Revised 13/06/2026
Accepted 07/07/2026

Keywords:

Driver Management
Pengembangan
Waterfall
PIECES
Blackbox Testing
Notifikasi
WhatsApp

ABSTRACT

Katadata's Driver Management System (DMS) previously facilitated structured driver and vehicle scheduling; however, it exhibited limitations in functionality, scheduling flexibility, and user interface design. This study aimed to analyze the existing system and develop an enhanced DMS to improve operational efficiency and usability. System requirements were identified using the PIECES framework, while the development process followed the Waterfall software development model. Several new features were implemented, including WhatsApp-based notifications, interactive maps for multipurpose operational support, enhanced scheduling based on location, fuel consumption, and vehicle status, as well as a driver performance evaluation module. Functional validation was conducted using Black Box testing to verify that all implemented features satisfied both functional and non-functional requirements without affecting the stability of the existing system. The results demonstrate that the enhanced DMS improves scheduling flexibility, increases the accuracy and accessibility of operational data, and provides more comprehensive information to support decision-making. Furthermore, the utilization of open-source libraries contributed to reducing development costs while maintaining system performance and scalability.

Corresponding Author:

Muhammad Bagir
Department of Information System,
Sekolah Tinggi Informasi Teknologi NIIT,
Jl. Asem Dua No. 22, Kel. Cipete Selatan, Kec. Cilandak Jakarta Selatan
Email: bagir.ui@gmail.com

1. INTRODUCTION

Katadata's Driver Management System (DMS) application has assisted in the digital scheduling and recording of drivers and vehicles, but its functionality and user interface remain limited. The company's evolving operational needs require the addition of instant notification features [1], flexible destination mapping [2], and driver performance evaluation. This study focuses on further development to address these issues and ensure that the system can more reliably support modern workloads.

Several studies on driver and vehicle management systems have been conducted previously. Eriyanto et al. [2] developed a web-based driver management system focused on basic scheduling and trip recording. However, the system has yet to integrate multi-channel notifications and driver performance evaluation. Saraswaha et al. [3] added notification features to the online attendance system; however, these were limited to mobile notifications without WhatsApp API integration and interactive mapping.

Commercial systems, such as Fleet Complete and Verizon Connect, provide fleet management with GPS tracking and real-time notifications. However, these systems have limitations in terms of customization for the specific needs of local companies and high implementation costs. Unlike similar systems, the DMS Katadata developed in this study offers a unique combination: (1) integration with the WhatsApp API for notifications familiar to local drivers, (2) cost-effective multi-destination mapping using OpenStreetMap, (3) an integrated performance review system, and (4) flexibility for customization to meet the needs of Indonesian companies.

The main contributions of this study are as follows: (1) development of a WhatsApp API integration framework for real-time notifications in the driver management system, (2) implementation of interactive multi-destination mapping using open-source libraries, (3) design of a driver performance evaluation system integrated with operational workflow, and (4) optimization of the system architecture using server-side processing to improve performance without increasing infrastructure costs.

The use of third-party libraries and APIs enables developers to integrate complex functions without building them from scratch [9]. In this study, Leaflet and OpenStreetMap provide interactive maps, OpenRouteService handles route calculation, and Venom-bot facilitates WhatsApp automation. Notifications via WhatsApp were chosen because drivers are more familiar with this instant messaging application compared to e-mail [1], [10]. In addition, the review module is designed to record users' assessments of driver performance so that management can continuously monitor service quality [11], [12]

Although the previous system managed basic scheduling, there were several gaps: notifications were limited to e-mail, address selection allowed only a single destination, and no performance feedback mechanism existed. This study aims to close these gaps by redesigning the interface, adding WhatsApp notifications, multi-destination map input, and a driver review module. Using the Waterfall approach and PIECES analysis, the development is expected to yield a more functional, efficient, and user-friendly DMS.

The novelty of this study lies in the multi-component integration that has not previously been implemented: (1) a native WhatsApp API framework for the Indonesian driver management system, (2) the combination of multi-destination mapping with real-time performance evaluation, (3) a hybrid architecture that optimizes performance without increasing infrastructure costs, and (4) a multi-layer testing methodology that ensures system validity from various aspects.

2. METHODS

This study employed a sequential Waterfall approach. The initial stage began with a literature review on driver management systems, API integration, and digital notification techniques, followed by direct observation of the legacy system to identify its functionalities and shortcomings. Based on the findings from these observations, requirements analysis was conducted using the PIECES model, which highlights performance, information, economics, control, efficiency, and service aspects. From this, functional specifications were formulated—such as the addition of WhatsApp notifications, multi-destination inputs with an interactive map, and a driver review module—as well as non-functional requirements related to performance, security, and usability.

System design includes the creation of a use case diagram, activity diagram, and ERD for the database structure. The interface is designed using an SB Admin 2-based wireframe to ensure consistency and responsiveness. Subsequently, implementation is carried out in Laravel 12 with integration of Leaflet/OpenStreetMap, OpenRouteService, Venom-bot for WhatsApp, and FullCalendar for schedule display.

Testing was conducted using several validation methods to ensure system quality. Black Box testing assesses functionality without inspecting the internal code, ensuring that the system responds to input according to specifications. White Box testing was conducted to test the internal code structure, path coverage, and logical flow. Load testing was also performed using Apache JMeter to ensure that the system can handle the daily operational data volume, which reaches 150–200 booking transactions per day with 50 concurrent users. The test results were then analyzed to ensure that all test cases ‘Pass’ and that there is no regression in existing functionality.

3. RESULTS AND DISCUSSION

3.1 PIECES Analisys

Based on the results of the observation, issues and problems in the predecessor application are grouped in the PIECES analysis matrix, as shown in Table 1.

Table 1. PIECES Analysis

Category	Current Issues
Performance	Overall, Laravel does not have specific performance issues, especially in handling the moderate request rate of this application; however, the version used is outdated.
	Not all tables were processed on the server side.
Information	Information regarding the vehicle and driver was incomplete.
	The admin dashboard page is not informative enough.
	The input destination address may be inaccurate.

Economy	There were no financial impediments.
Control	Access restrictions on certain pages can still be bypassed, for example, by accessing the *url* directly.
	Passengers and destination addresses were not recorded consistently.
	Driver performance was not monitored.
	Google Calendar events do not change, even if update or cancellation schedule emails are sent.
Efficiency	The data display during scheduling was inefficient.
	The travel schedule report is prepared separately.
Service	Drivers are not accustomed to using emails to receive notifications.
	The requester does not receive clear information about the driver, such as their reputation, reliability, or experience.
	The page layout is overly plain.

Referring to the explanation of the scope definition previously presented in Table 1, the next step is to outline the problems in Table 2, so that solutions for each issue can be clearly identified.

Table 2. Description of Problems

Problem	**Penyebab**	**Solution**
Outdated framework version.	The framework has undergone several changes since the initial development of the application.	Rewrite the codebase from scratch and establish writing standards for better organization.
The page load time was relatively slow.	A large amount of data is managed directly and simultaneously.	Increase the number of asynchronous data calls.
The page interface is unappealing.	The application development did not undergo a robust page design process.	Redesigning the page layout and utilizing available UI templates.
Incomplete information.	The previous application was not designed to comprehensively and relevantly accommodate certain information.	Data management was further developed for drivers and vehicles.
Poorly controlled travel logging.	There are no clear guidelines or standards for recording journeys when an application is built.	The new system provides passenger logging, direct address selection using maps, and fuel consumption.
Travel notifications to drivers were not communicated effectively.	The application only supports email delivery, whereas in practice, drivers are not accustomed to using email.	Implementation of WhatsApp as an alternative medium for travel notifications
The schedule saved in the Google Calendar is not updated when there are changes or cancellations.	New invitations are not included in the update and cancellation emails for trips.	Embedding invitations into emails with the correct configuration.
Driver performance was not monitored.	The previous application was not designed to accommodate the monitoring of driver performance.	Implementation of a driver review system for every trip.

3.2 System Needs Analysis

Functional Requirements:

1. The system must be able to manage travel schedules in an integrated manner with information related to the drivers, vehicles, and destinations.
2. The system must be capable of recording and managing passenger data, thereby enabling the reselection of scheduling subsequent trips.
3. The system must provide a destination address input that uses an online map with the option to add multiple destinations simultaneously.
4. The system must be able to identify the status of vehicles and drivers that have already been scheduled to prevent unavailability of vehicles or drivers.
5. The system must support asynchronous dynamic data processing using AJAX, as in tables, dashboards, and other data inputs.

6. The system must allow the delivery of schedule notifications via WhatsApp to the registered driver's phone number.
7. The system must be able to store and manage additional driver data, such as work experience and leave schedules.
8. The system must store and display company vehicle data with more complete information, such as fuel type and vehicle image.
9. The system can generate a travel report from the Travel List page.
10. The system can generate driver-review reports.

Non-Functional Requirements:

1. The system must be rewritten using the latest version of the framework with better code-writing standards to support maintainability and scalability.
2. The application display page must be designed using CSS templates to enhance the aesthetics and user experience.
3. User authorization must be reviewed and tested to ensure system security and appropriate access.
4. The system must have adequate performance to handle large and dynamic data volumes without reducing the access speed.
5. The system must provide ease of navigation with an intuitive user interface, including the booking page and dashboard.
6. The system must support the addition of information related to drivers and vehicles to improve transparency and decision-making.
7. The system must accurately display data.

3.3 System Design

The next step was to conduct the design based on the previous analysis to satisfy the new requirements.

Figure 1. Use Case Diagram

In this system, there are three types of users, each with their own roles. Admin is responsible, as the data manager, from managing users to handling travel histories through the booking feature. Requester is a user who needs to book a car for travel purposes. Meanwhile, Driver serves as the driver who must monitor their travel schedule and carry out the task of transporting passengers or goods as requested.

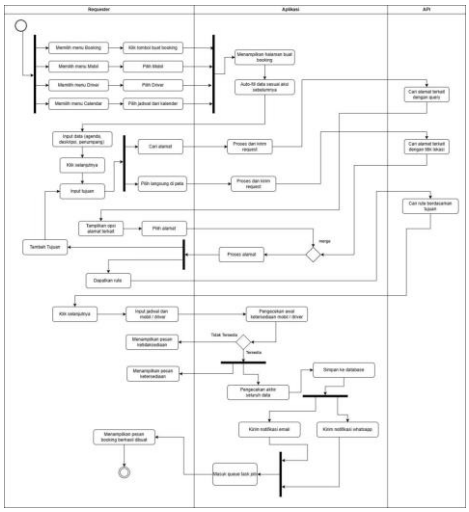


Figure 2. Car Booking Activity Diagram

Users can make bookings through various pages, either manually from the booking log page or directly via shortcuts on the table, such as selecting a specific driver from the Driver page. The booking form consists of three sections: schedule information, destination information, and selection of driver or vehicle along with the departure schedule. After the data is reviewed and saved, a notification is sent to the driver's application task list.

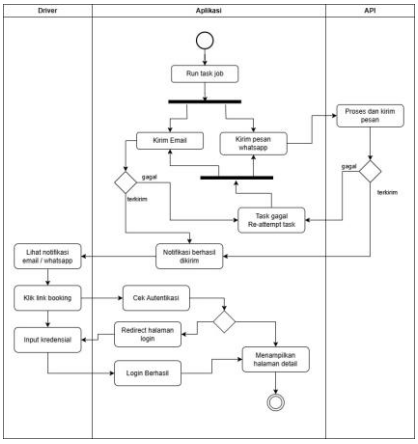


Figure 3. Activity Diagram for Sending Notification and Viewing Booking

The application continuously checks and executes whenever there are jobs in the task list queue, and at that moment the application sends email and WhatsApp notifications to the drivers. If successfully sent, the driver will receive the notification and can access the details of their respective trip assignments.

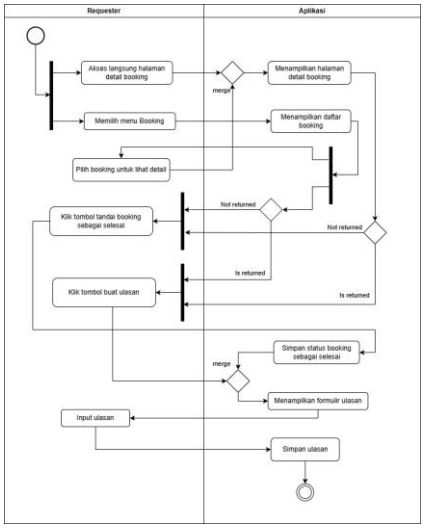


Figure 4. Activity Diagram for Writing a Review

This activity begins once the trip has been declared complete. To mark it as complete, the user can either directly access the shortcut in the booking log table or go to the booking detail page and mark the booking as finished. Subsequently, the user is prompted to enter a review in the form provided. If they have not had the chance or for other reasons, the user can reopen it later to submit their review.

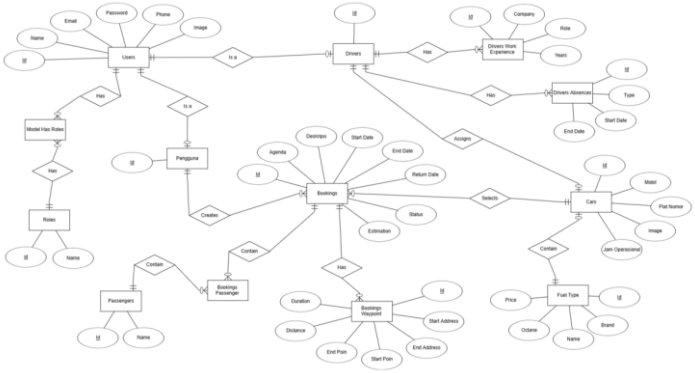


Figure 5. ER Diagram Conceptual Data Model

This model describes a transportation booking management system based on users, drivers, and vehicles. The main entities consist of Users, Drivers, Cars, Bookings, and Passengers, with several supporting entities such as Roles, Drivers Work Experience, Drivers Absences, and Fuel Type.

3.4 System Implementation

Proceed to the stage of realizing what was designed in the previous stage. The following are some examples of the system interface implementation results.

1. Dashboard Interface

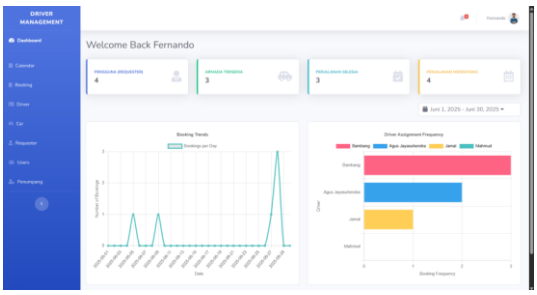


Figure 6. Admin Dashboard

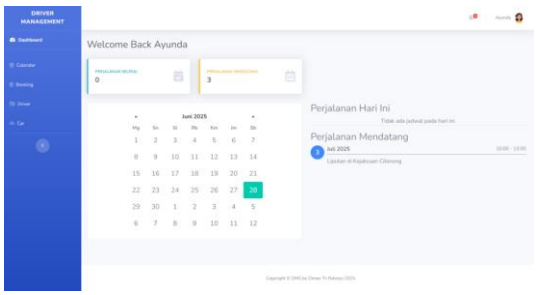


Figure 7. Requester and Driver Dashboard

This is the new dashboard interface display for each type of user. Each user has different information needs and fast information access.

2. Booking Interface

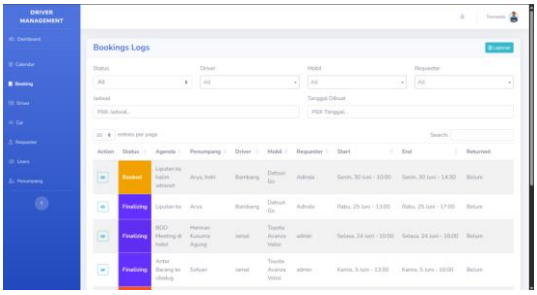


Figure 8. Booking List (Admin)

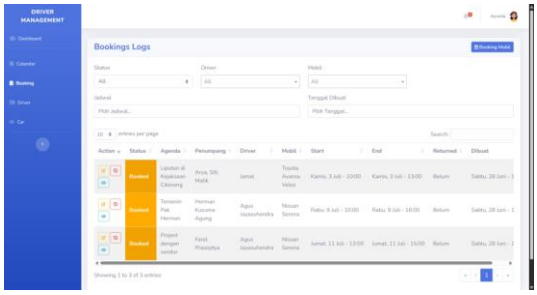


Figure 9. Booking List (Requester)

Each type of user has different data access rights; therefore, the data displayed will be adjusted according to whom it is shown for, along with the actions the user can perform with that data.

3. Make a Booking

This page has undergone a major overhaul to meet new data requirements, such as addresses and passengers. Therefore, for more effective scheduling, the form has been divided into three sequential sections: agenda, destination, and car.

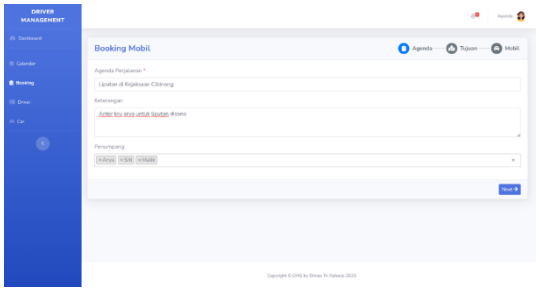


Figure 10. Stage One Booking (Agenda)

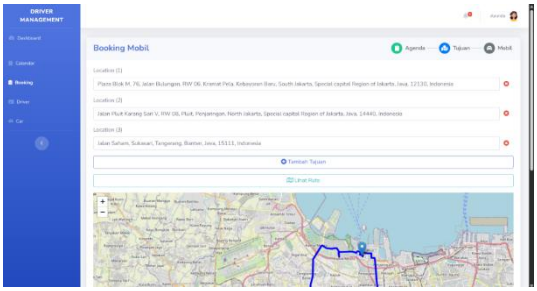


Figure 11. Second Stage Booking (Destination)

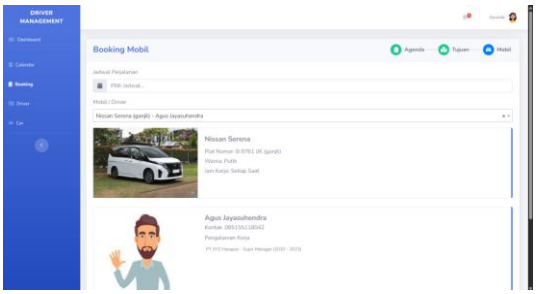


Figure 12. Stage Three Booking (Car)

4. Notification

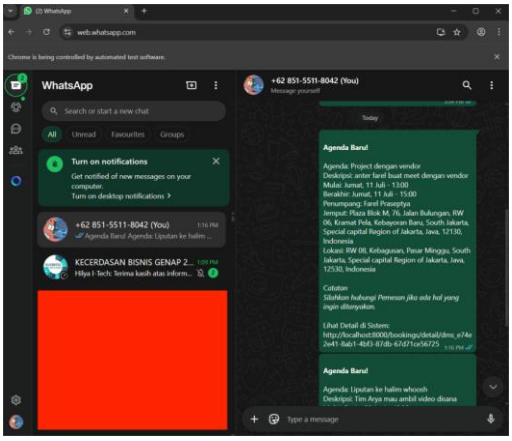
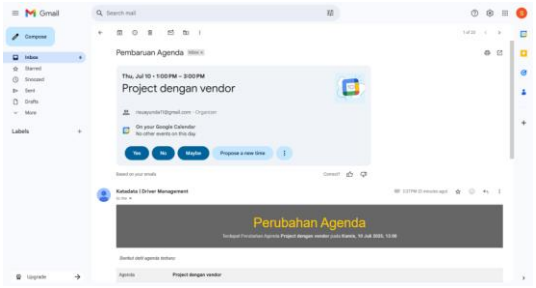


Figure 13. WhatsApp Notification Message

This is a sample message that was automatically sent by the system to the driver regarding a new agenda.



Gambar 14. Notifikasi Perubahan Jadwal Pada Email

This is an example of an agenda change email received by the driver, along with Google Calendar Event information, which was not available in the previous system.Comparative performance analysis shows a significant improvement from the old system to the new system. The average response time decreased from 2.4 seconds to 0.8 seconds (66% improvement). Memory usage was reduced by 35% with the implementation of server-side processing. User satisfaction increased from 3.2/5 to 4.7/5 based on the UAT survey. The system can handle up to 50 concurrent users without performance degradation, compared to the old system, which could only support 20 concurrent users.

3.5 System Testing

Table 3. BlackBox Testing

Case ID	Test Case	Expected Results	Status
TC101	Admin login	Successfully logged in and reached the Admin Dashboard interface.	Pass
TC102	Requester login	After successfully logging in, the Requester Dashboard interface was accessed.	Pass

TC103	Driver login	Successfully logged in and reached the Driver Dashboard interface.	Pass
TC104	Menu navigation	Menu and all action buttons or links according to each user's access rights.	Pass
TC201	User list	All registered user types are displayed in the table, and the filter shows the data according to the selected criteria.	Pass
TC202	Add admin user	The user list page is returned, and a notification that it was saved successfully is received. The newly saved user appears on the list.	Pass
TC203	Edit user	Return to the user list page and receive a notification that it was saved successfully. User data has changed.	Pass
TC204	Delete user	The updated user table list and deleted users disappear.	Pass
TC205	Add requester role to admin	The role requester appears in the list of roles owned by the selected admin user.	Pass
TC206	Remove the requester role from admins who have the requester role	The role requester disappears from the list of roles belonging to the selected admin user.	Pass
TC207	Profile interface	User-owned information is visible.	Pass
TC208	Change profile photo	Profile photo changed.	Pass
TC209	Change password	A notification appears indicating that the password has been changed.	Pass
TC301	Driver list	All users with the driver role are listed in table.	Pass
TC302	Add driver user	Redirected to the driver edit page. Previously saved driver data are automatically filled in.	Pass
TC303	Edit user driver	Return to the driver list page and receive a notification that the data has been saved successfully. The driver data has changed.	Pass
TC304	Add work experience	A notification that the data has been saved will appear.	Pass
TC305	Edit work experience	A data saved notification will appear.	Pass
TC306	Delete work experience	A notification that data have been deleted will appear and disappear from the list.	Pass
TC307	Import work experience	Data has been saved and the list will change.	Pass
TC308	Add driver absence	A notification that the data has been saved will appear.	Pass
TC309	Edit driver absence	Data saved notification will appear.	Pass
TC310	Delete driver absence	A notification that the data has been deleted will appear and be removed from the list.	Pass
TC311	Driver absence import	Data has been saved and the list will be updated.	Pass
TC312	Delete user driver	Updated driver table list and deleted users disappear.	Pass
TC313	Driver detail	Redirected to the driver detail page, and detailed information related to the driver is displayed, such as the driver, car, and driver reviews.	Pass
TC401	Requester list	All users with the requester role are included in the table list.	Pass
TC402	Add requester user	Return to the Requestor List page and receive a successful save notification. The new requester user now appears in the table list.	Pass
TC403	Edit user requester	Return to the requester list page and receive a notification that it was saved successfully. The requester data has changed.	Pass
TC404	Delete user requester	Updated requester table list and users who were deleted disappear.	Pass
TC501	List of cars	All cars are displayed in the table list.	Pass

TC502	Add car	Return to the car list page and receive a notification that it has been successfully saved. The newly added car appears in the table list.	Pass
TC503	Edit car	Return to the car list page and receive a notification that it was saved successfully. Car data has changed.	Pass
TC504	Delete car	Updated requester table list and deleted users disappear.	Pass
TC505	Car detail	Directed to the car detail page and detailed information related to the displayed car, such as driver, car, and driver reviews.	Pass
TC506	Update fuel price list	Notification: The data has been updated, and if there are any price changes, the price will be adjusted.	Pass
TC601	List all bookings	All booking lists are displayed in table. The data displayed matches the filter used.	Pass
TC602	Booking requester list	The displayed booking list belongs only to the requester. The data displayed matches the filters used.	Pass
TC603	Driver booking list	The displayed booking list belongs only to the drivers. The data displayed correspond to the filters applied.	Pass
TC604	Make a booking	a) There were no error notifications during data entry, and it was possible to proceed to the final stage. b) Getting the route from the input location. c) Displays detailed information related to the selected car and driver. d) Receive notifications that the car is available or unavailable on the selected schedule. e) Redirected to the booking list page and receive a notification that the booking was successfully created. f) Notification sent.	Pass
TC605	Make a booking from the driver module	a) There were no error notifications during data entry and it could proceed to the final stage. b) Getting a route from the input location. c) The selected car is automatically selected and displays detailed information related to the chosen car and the driver. d) Receive notification that the car is available or not available at the selected schedule. e) Redirected to the booking list page and receive a notification that the booking was successfully created. f) Notification sent.	Pass
TC606	Make a booking from the car module	a) There were no error notifications during data entry, and it was possible to proceed to the final stage. b) Getting directions from the entered location. c) The selected automatic car and display detailed information related to the selected car and driver. d) Receive notifications that the car is available or unavailable on the selected schedule. e) Redirected to the booking list page and received a notification that the booking had been successfully created. f) Notification sent.	Pass
TC607	Edit booking	a) No error notifications were observed during data entry and saving. b) Receive a notification that the car is available or not on the selected schedule. c) Redirected to the booking list interface and receives a notification that the booking has been successfully created. d) Notification sent.	Pass
TC608	Edit booking (car replacement)	a) No error notifications were observed during data entry and saving. b) Receive notifications that the car is available or unavailable on the selected schedule. c) Redirected to the booking list page and received a notification that the booking has been successfully created. d) Notification was sent to both drivers, the previous and new drivers.	Pass
TC609	Cancel booking	A notification that the booking cancellation was successful will appear, and the booking status will change.	Pass

TC610	Detail booking	The booking detail page and all information related to the booking will be displayed.	Pass
TC611	Return booking	A notification indicating a successful booking return will appear, and the booking status will change. After that, a popup will appear to fill in a review of the agenda.	Pass
TC612	Review booking	Notification that the review has been successfully saved and that there is no longer a button to submit a review.	Pass
TC613	Fuel consumption	Notification that fuel expenditure data have been saved, and the information will be listed in the fuel expenditure section.	Pass
TC614	Calendar booking	The large calendar fills the entire content area of the interface, and the booking list is displayed according to the schedule.	Pass
TC615	Make a booking from the calendar	a) There were no error notifications during data entry, and it was possible to proceed to the final stage. b) Obtaining a route from the entered location. c) Automatically selected schedules. d) Receive notifications of whether a car is available or unavailable on the selected schedule. e) Redirected back to the booking list page and receive a notification that the booking has been successfully created. f) Notifikasi terkirim.	Pass
TC616	Export booking	Fill in the data in the file according to what is shown in the Booking List table.	Pass
TC701	Passenger list	All passengers listed in the passenger list table.	Pass
TC702	Add passenger	New passenger notifications have been saved and added to the table list.	Pass
TC703	Edit penumpang	The passenger data notification has been saved, and the table list has also changed.	Pass
TC704	Delete penumpang	The passenger notification has been deleted and removed from the table list.	Pass
TC801	Check email notifications	a) Receive notification alerts about bookings, whether for new agendas, changes, or cancellations. b) Redirected to the Google Calendar website, and the agenda listed in the email was recorded.	Pass
TC802	Check WhatsApp notifications	Receive notifications regarding bookings, whether new agendas, changes, or cancellations.	Pass
TC803	Cek notifikasi terkini	Several recent notifications that have not been viewed are displayed in the list. If there are no notifications, a message appears stating that there are no unseen notifications.	Pass
TC804	List of all notifications	The user is redirected to the notifications list page, where all previously received notifications are displayed. Notifications can be clicked to display complete information or be directed to relevant matters.	Pass
TC805	Mark notification as read	All selected notifications will change their appearance to seen.	pass
TC806	Mark notification as unread	All selected i notifications will change their appearance to unread.	Pass

Table 4. White Box Testing

Case ID	Module/Function	Tested Pathway	Expected Results	Status
WB01	createBooking()	Input valid → data validation → check vehicle availability → save booking → send notification	Data is stored in the database and notifications are added to the task list queue	Pass
WB02	createBooking()	Input valid → car not available on schedule → display error message	The system rejects the booking and displays a message that the car is not available	Pass
WB03	sendWhatsappNotification()	New booking → message format → send via Venom-bot API → successful response	WhatsApp sent to the registered driver number	Pass

WB04	sendWhatsappNotification()	API failed → retry mechanism → record error log	The system attempts to resend and the error is recorded in the log	Pass
WB05	checkCarAvailability()	Schedule conflicts with active booking → return false	Function returns unavailable status	Pass
WB06	checkCarAvailability()	Schedule does not conflict → return true	Function returns available status	Pass
WB07	storeReview()	Input valid rating and comment → save to database → update driver's average rating	Saved reviews and average driver rating updated	Pass
WB08	storeReview()	Booking belum berstatus returned → tolak input review	The system rejects and displays a message that the booking is not yet complete	Pass
WB09	generateRoute()	Multi-purpose valid → call OpenRouteService API → return route polyline	Route successfully calculated and displayed on the map	Pass
WB10	generateRoute()	Address not found → return error from API	The system displays an invalid address message	Pass
WB11	exportBooking()	Filter data → query database → generate file Excel	Excel file contains data according to the applied filter	Pass
WB12	serverSideProcessing()	Request dengan parameter pagination dan filter → query optimized → return JSON	Data is returned as per the parameters with a response time < 1 second	Pass

White Box testing was conducted on 12 critical system logic paths. All test cases resulted in a Pass status, demonstrating that the internal code structure operated according to the designed flow without any dead code or logical errors in the main functions.

Table 5. Load Testing (Apache JMeter)

Scenario	Number of Concurrent Users	Average Response Time	Throughput (req/s)	Error Rate	Status
Light Load	10	0,4 detik	25	0%	Pass
Normal Load	25	0,6 detik	42	0%	Pass
Peak Load	50	0,8 detik	78	0%	Pass
Stress Load	75	1,5 detik	95	2,1%	Pass
Overload	100	3,2 detik	110	8,5%	Warning

Load Testing was conducted using Apache JMeter with a tiered scenario. The system is able to handle up to 50 concurrent users (Peak Load) without errors, and the response time remains under 1 second. Under Stress Load conditions (75 users), the error rate is still within the tolerance limit (< 5%). Significant degradation only occurs in the Overload scenario (100 users), which exceeds daily operational capacity. These results demonstrate that the system is sufficiently reliable to handle Katadata's operational volume, which averages 150–200 transactions per day with a maximum of 50 active users at the same time. Further development of the DMS has added WhatsApp notification functionality, interactive map integration, and performance review features, while also improving the user interface. The implementation of server-side processing and the latest frameworks has enhanced performance without increasing operating costs. Multi-method testing (Blackbox, White Box, UAT, Load Testing) showed all test cases achieved 'Pass' status with a 66% performance improvement and 47% user satisfaction.

For integration into production, incremental data migration is recommended, configuration of cron jobs for automatic notifications, real-time performance monitoring using tools such as New Relic, and regular system backups. Further research can be directed towards: (1) Implementation of machine learning for vehicle demand prediction, (2) Integration of IoT sensors for real-time vehicle condition monitoring, (3) Development of a mobile application for drivers and requesters, and (4) Implementation of blockchain for a more secure audit trail.

DAFTAR PUSTAKA

[1] D. Bayu Anjasmara, M. A. Rosid, and A. Eviyanti, "Implementasi Fitur Notifikasi Whatsapp API pada Sistem Manajemen Tugas Akhir," Physical Sciences, Life Science and Engineering, vol. 1, no. 2, pp. 1–14, 2023, doi: 10.47134/pslse.v1i2.197.

[2] S. E. Eriyanto, A. Eviyanti, A. S. Fitrani, and U. Indahyanti, "Rancang bangun Manajemen driver berbasis web di PT. XXX," JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika), vol. 9, no. 1, pp. 100–113, Feb. 2024, doi: 10.29100/jipi.v9i1.4358.

[3] I. P. S. Saraswaha, A. P. Kharisma, and A. Pinandito, "Pengembangan Lanjut Aplikasi berbasis Android Sistem Presensi Onlinedengan Penambahan Fitur Notifikasi dan Fitur Permohonan Ijin atau Cutidi PT. Bintang Mas Glassolutions," Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, vol. 7, no. 4, pp. 2010–2019, Apr. 2023, [Online]. Available: https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/12641

[4] H. Miftakhuddin and H. Thamrin, "Pengembangan Modul Pengelolaan Studi Lanjut pada Sistem Informasi Sumber Daya Manusia," Just IT : Jurnal Sistem Informasi, Teknologi Informasi Dan Komputer, vol. 12, no. 2, pp. 27–33, Aug. 2022, doi: 10.24853/justit.12.2.

[5] D. Siswanto, Y. Darmayunata, Z. Zamzami, and B. Febriadi, "Pengembangan lanjut Sistem Informasi Pelayanan dan Pendataan UMKM Provinsi Riau (Kasus Dinas Perindustrian, Perdagangan, Koperasi Usaha Kecil dan Menengah Provinsi Riau)," Brahmana : Jurnal Penerapan Kecerdasan Buatan, vol. 5, no. 1, pp. 43–50, Dec. 2023, doi: 10.30645/brahmana.v5i1.276.

[6] M. I. Hossain, "Software Development Life Cycle (SDLC) Methodologies for Information Systems Project Management," International Journal for Multidisciplinary Research, vol. 5, no. 5, 2023, doi: 10.36948/ijfmr.2023.v05i05.6223.

[7] M. K. Argaputri, N. R. Fadlilah, H. Setyowati, and M. A. Yaqin, "Analisis Model PIECES dalam Perancangan Sistem Informasi Meeting Proyek Menggunakan Metode FAST," ILKOMNIKA - Lembaga Penelitian Dan Pengabdian Masyarakat, vol. 5, no. 1, pp. 59–70, Apr. 2023, doi: 10.28926/ilkomnika.v5i1.418.

[8] L. Setiyani, Y. Rostiani, and T. Ratnasari, "Analisis Kebutuhan Fungsional Sistem Informasi Persediaan Barang Perusahaan General Trading (Studi Kasus : PT. Amco Multitech)," Owner, Riset & Jurnal Akuntansi, vol. 4, no. 1, pp. 288–295, 2020, doi: 10.33395/owner.v4i1.205.

[9] G. Jackson, "What is API integration?," IBM, Feb. 07, 2025. https://www.ibm.com/think/topics/api-integration

[10] Z. Istifazah, "Notification Mobile based Pelaporan Data Pokok Pendidikan (Dapodik) Guru Sekolah Menengah Atas (SMA) Kota Bandar Lampung," Institut Informatika Dan Bisnis Darmajaya, 2020. [Online]. Available: http://repo.darmajaya.ac.id/3170/

[11] J. J. Cpr, S. Sundari, and M. Pakpahan, "Pentingnya feedback (Umpan balik) konstruktif di dalam lingkungan kerja," eBisnis Manajemen, vol. 2, no. 1, pp. 147–159, Feb. 2024, doi: 10.59603/ebisman.v2i1.349.

[12] D. Giamos, O. Doucet, and P.-M. Léger, "Continuous Performance Feedback: Investigating the Effects of Feedback Content and Feedback Sources on Performance, Motivation to Improve Performance and Task Engagement," Journal of Organizational Behavior Management, vol. 44, no. 3, pp. 194–213, 2025, doi: 10.1080/01608061.2023.2238029.