International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 01, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

Manake Media Equipment Rental System with Payment Gateway and Business Process Optimization

[1st]Fikri Mulya Rachmat, [2nd]Cholid Mawardi, [3rd]Yudha Pradana
Multimedia Engineering Technology
Politeknik Negeri Media Kreatif Jakarta, Indonesia
[1]2290343040@polimedia.ac.id, [2]cholid@polimedia.ac.id,[3]yudha.pradana@polimedia.ac.id

Abstract**—***Media equipment rentals are time-sensitive because a single item can only be booked within a specific date range and requires payment confirmation before the unit is prepared. Manual handling through WhatsApp conversations and bank transfers often creates obstacles such as slow confirmation, schedule conflicts, and non-standardized transaction records. This study designs and implements "Manake", a web-based media equipment rental management system that integrates the Midtrans payment gateway (Snap) and webhooks to synchronize payment status in real time. Development follows the Waterfall model, covering requirements analysis, design, implementation, and testing. The system is built using Laravel 12 (PHP 8.5.2), a Tailwind CSS interface through Vite, and a managed PostgreSQL database on Supabase. Business rules are embedded directly into the system, including a one-day operational buffer before and after the rental period, a reschedule policy limited to one time before pickup with no refunds, percentage-based late-return penalties calculated per order, and minimum damage fees calculated per item. Evaluation is conducted using black-box testing on core scenarios aligned with the business flow (catalog, cart, checkout, payment, invoice, and admin operations). The results show that all main functions run according to specification, and the webhook integration is able to update transaction status automatically through signature validation and event deduplication mechanisms.*

Keywords : *website, rental management system, payment gateway, business process optimization, booking schedule*

I. INTRODUCTION

In micro-scale and niche businesses, services often grow from local communities. In the case of Manake, the service idea emerged from a common habit in a circle of friends: asking where to rent media equipment, which later developed into a commercial service. However, when operations are still managed through chat and manual record-keeping, response speed, transaction traceability, and consistency of operational rules become difficult to maintain.

The context of digital adoption in Indonesia further strengthens the urgency for responsive web-based services. An APJII survey reported that Indonesia had 221.56 million internet users with a penetration rate of 79.5% in 2024, making digital channels a primary medium for interaction and transactions. [2]

On the payment side, Bank Indonesia recorded that QRIS had reached 57 million users and 39.3 million merchants by the first semester of 2025, indicating the normalization of digital payments across various segments, including MSMEs. Integrating a payment gateway into a rental system has the potential to accelerate payment confirmation and reduce the workload of manual verification. [3]

Previous studies have shown that web-based business process integration can improve operational efficiency and data consistency. Mawardi et al. emphasized the importance of integrating order and payment flows in an integrated web system to enhance business process efficiency. [1]

In the context of Midtrans payment integration, several studies have implemented Midtrans Snap to accelerate transaction processing and synchronize payment status in web-based applications. Nevertheless, many implementations focus on e-commerce or digital top-ups, so rental-specific requirements—such as schedule overlap

validation, an operational buffer, limited rescheduling, and late-return or damage penalties—are often not implemented as consistent system rules. [4], [5], [6]

Studies on web-based rental systems across different domains point to similar issues, namely dependence on manual records that are prone to data loss and service delays. In payment-integrated rental systems, research on Graha Mustika Hall implemented a rental system integrated with a payment gateway to speed up transactions, while another study on electronic equipment rentals highlighted improved data accuracy and operational efficiency through digitalization. In addition, a black-box testing approach based on state transitions has been shown to be effective for ensuring web function stability under status transition scenarios. The literature also notes that the choice of development model (e.g., Waterfall or Agile) affects team communication patterns and outcome quality, so method selection should be aligned with the project context[7]–[11], [14],[15].

Based on these gaps, this study contributes by: (1) designing a web-based media equipment rental system that models stock and date-range availability with an operational buffer; (2) integrating the Midtrans Snap payment gateway with webhooks and signature validation for automatic payment status updates; (3) implementing an admin operations module for rental status control, additional fees, and notifications; and (4) conducting functional evaluation using black-box testing on core scenarios aligned with the business flow.

II. RESEARCH METHODS

2.1 Research Type and Data Sources

This study is a software engineering study that produces an information system artifact for rental management. Data sources are obtained through observation of the business process, documentation review (internal invoices/receipts and operational rules), and functional testing of the system. This study does not use respondents or questionnaires. The evaluation focuses on whether system functions match the defined specifications.

2.2 System Development Method (Waterfall)

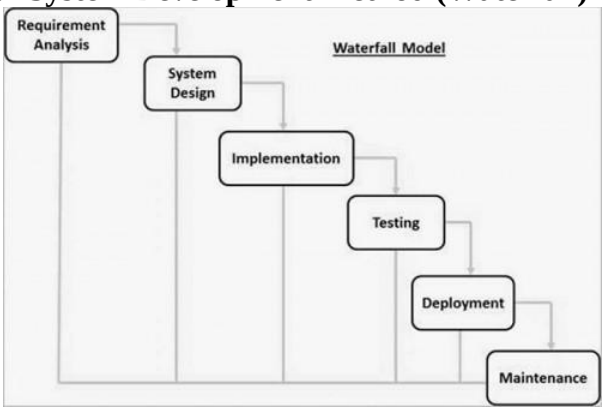

Figure 1. Stages of system development using the Waterfall model.

Figure 1 shows that the development process follows the Waterfall model through sequential stages. In this study, the Waterfall model consists of: (1) requirements analysis, (2) design, (3) implementation, and (4) testing. The requirements analysis produces a list of core features, including an equipment catalog, date- and quantity-based availability validation, cart and checkout, Midtrans payment integration, automatic invoicing, limited rescheduling, and an admin panel for operational and content management.

2.3 Architecture, Implementation Environment, and Payment Integration

The system architecture applies a three-layer approach: (a) a client browser for the user interface, (b) a web application (Laravel) for business logic, and (c) a managed PostgreSQL database (Supabase) for data storage. The payment component uses Midtrans Snap as the interface for selecting payment methods and a webhook callback to receive transaction status notifications. The callback endpoint is available on the routes POST /midtrans/callback and POST /api/midtrans/callback.

2.4 Tools and Technologies Used

The implementation uses Laravel 12.49.0 on PHP 8.5.2, the midtrans/midtrans-php package for Snap integration, and barryvdh/laravel-dompdf for generating invoice PDFs. The front end is built with Vite and Tailwind CSS, with Alpine.js used for lightweight client-side interactions.

2.5 Business Rules and Operational Policies

The main business rules implemented as system rules include: (1) a one-day operational buffer before and after the rental period for preparation and maintenance; (2) rescheduling is allowed a maximum of one time before the item is picked up, only if the new date is available, with no refund; (3) late-return penalties are calculated per order as a percentage of the daily rental cost (3 hours or more: 30%, 6 hours or more: 50%, more than 9 hours: 100%); (4) damage fees are calculated per item with a minimum value of 50% per item based on the admin's assessment; and (5) lost items are charged at 100% of the unit price.

2.6 Testing Method (Black Box)

Testing is conducted using a black-box approach to verify functional compliance with business requirements without examining the internal code structure. Test scenarios are prepared following the core transaction flow (catalog, date selection, cart, checkout, Midtrans payment, invoice) and the operational flow (updating rental status, rescheduling, and additional fees or damage fees). Payment testing is performed in the Midtrans sandbox environment. Transaction status is received through the webhook endpoint and verified using an SHA-512 signature.

III. RESULT AND ANALYSIS

In this section, the discussion focuses on the design and implementation results of the Manake system as a solution to the time-sensitive nature of media equipment rentals. In this business model, a single item can only be rented within a specific date range, so schedule conflicts must be mitigated from the date selection stage through checkout. At the same time, payment certainty becomes a prerequisite before the unit is prepared, which means the system should reduce reliance on manual verification. Therefore, Manake is designed as a web-based system that integrates schedule locking, automated cost calculation, and payment through a payment gateway, so the rental flow becomes more standardized and traceable.

The implementation results in Section III are presented step by step. First, the system and database design are described as the foundation for separating catalog entities from transaction entities to support future expansion. Second, the availability validation mechanism is explained, including date overlap checking and a one-day operational buffer before and after the rental period. Third, the user module is presented with an emphasis on a low-friction transaction experience, starting from browsing the catalog without login through checkout and invoice issuance. Fourth, the Midtrans Snap and webhook integration is discussed, including signature validation and duplicate event prevention to ensure consistent payment status. Finally, the operational admin module is described, covering monitoring, rental status management, additional charge handling (late return penalties charged per order and damage fees charged per item), and notification delivery to support more structured operational execution.

3.1 System and Database Design

Figure 2. Use case diagram of the Manake Rental system.

The use case diagram in Figure 2 outlines the main interactions between actors and the Manake system. The Customer/User actor covers the end-to-end rental journey: registering or logging in, browsing the catalog, opening equipment details, checking availability, adding items to the cart, completing checkout, making payments, and viewing invoices. The Operational Admin actor manages operational control by monitoring and updating rental status, maintaining equipment and category data, applying additional charges when required, and sending notifications to users. The payment gateway, Midtrans, supports the Snap payment flow and delivers transaction

status updates to the system through webhook callbacks, ensuring payment information is synchronized with operational processing. This integration enables real-time confirmation, reduces manual checks, and improves consistency for customers and administrators.

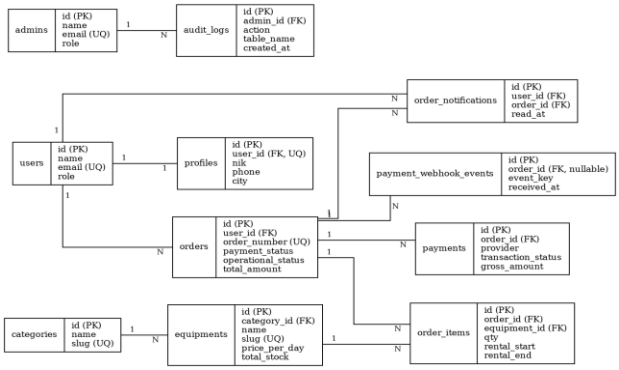

Figure 3. Database design (ERD) and relationships among main tables.

Figure 3 illustrates a core data model that separates catalog entities from transaction entities to support future expansion. Catalog data is stored in CATEGORIES and EQUIPMENTS, where EQUIPMENTS maintains attributes such as name, slug, total_stock, daily price, and readiness status. Transaction data is stored in ORDERS, which records the rental period, cost components (subtotal, 11% VAT, total amount), and both payment and operational status. Each rented item is captured in ORDER_ITEMS using references such as order_id and equipment_id, along with quantity and pricing details. Payment processing is recorded in PAYMENTS, while webhook deliveries are logged in PAYMENT_WEBHOOK_EVENTS to support signature verification and idempotency. Operational communication is supported through ORDER_NOTIFICATIONS, and administrative actions can be traced through ADMINS and AUDIT_LOGS when required.

The database uses managed PostgreSQL on Supabase and is accessed through the pgsql driver configuration. Using a managed database reduces infrastructure workload such as backups and patching. It also supports scaling through connection settings, indexing, and optional read and write separation when needed. This reduces complexity and supports secure connections as demand and traffic grow.

3.2 Availability Validation and Operational Buffer

Equipment availability is calculated based on overlap between the requested rental date range and active transactions. The system extends the rental period by adding a one-day buffer at the start and end to prevent schedules from becoming too tight. For each equipment item, the total number of units in use within the extended range is calculated by aggregating ORDER_ITEMS under ORDERS whose status is not failed or expired and whose rental lifecycle has not been completed. Checkout is rejected when (total_stock minus units_in_use) is less than the requested quantity.

The availability validation procedure follows these steps. First, the system extends the rental period by one day before and one day after. Second, it calculates how many units are currently booked within any overlapping date ranges. Third, available stock is derived from total stock minus units in use. Fourth, checkout is rejected if available stock is smaller than the requested quantity.

3.3 User Module and Transaction Experience

The interface is designed to minimize friction. Users can browse the catalog, search, and view recommendations without logging in. Login is required only when adding items to the cart or proceeding to payment. In addition, the rental profile stores validated identity and contact information so that future transactions become faster and reduce data entry errors. Figure 4 shows the catalog and equipment detail interface, including the blocked usage schedule and the operational buffer that helps users understand availability before checkout.

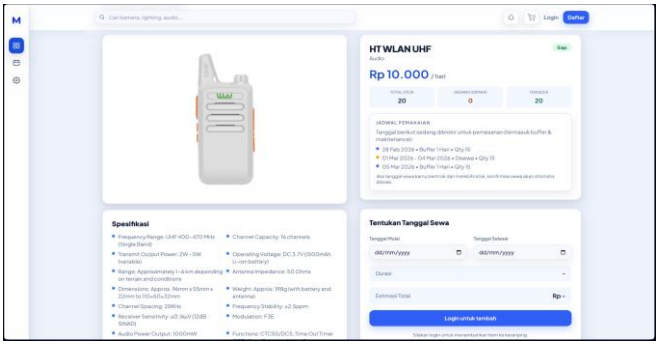

Figure 4. Catalog and equipment detail page, including the usage schedule and buffer.

The rental cart automatically calculates the daily subtotal, the estimated total based on rental duration, and 11% VAT before the user proceeds to payment. During checkout, the system creates a Midtrans Snap transaction and displays a payment pop-up so users can select a payment method, such as a virtual account. This reduces friction compared to manual transfer verification because payment status does not depend on proof sent through chat. [4] Figure 5 shows the cart view and the cost breakdown presented to the user, including subtotal, 11% VAT, and the final total before payment is confirmed.

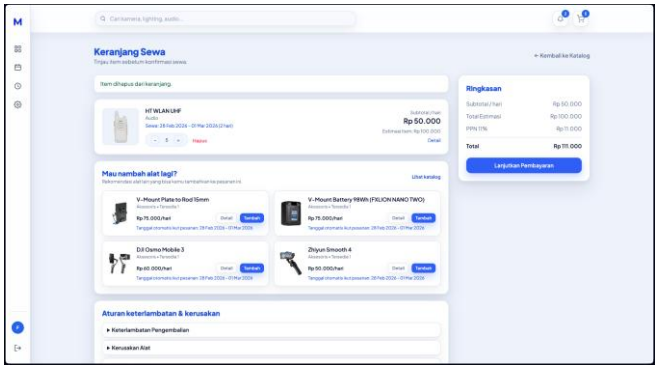

Figure 5. Rental cart with subtotal, 11% VAT, and total calculation.

The reschedule feature is available on the order detail page and is active only before the operational status changes to “Item Picked Up”. The system limits rescheduling to a maximum of one time, keeps the same rental duration and item quantity, and re-applies the buffer and stock availability rules to the new dates. The no-refund policy helps maintain cash flow stability and reduces the risk of unilateral cancellations after the schedule has already been blocked.

In line with implementations in several web-based applications, using Midtrans as a payment gateway helps standardize the payment flow and reduce manual verification processes, including for web-based shops and MSMEs. [12], [13]

3.4 Payment Integration and Status Consistency

At the system level, the payment gateway is treated as the primary reference for payment status. Each local transaction record (ORDERS and PAYMENTS) is synchronized with Midtrans status through callbacks. This approach reduces differences in interpretation between customers and admins because “Paid”, “Failed”, “Expired”, and “Refunded” are derived directly from Midtrans transaction statuses. As shown in Figure 6, Manake uses the Midtrans Snap interface to let users select a payment method, including virtual accounts, while the system relies on callback notifications to keep payment status consistent. A similar approach is widely used in Midtrans Snap implementations in other domains to ensure automatic status updates and reduce admin workload. [5], [6], [12]

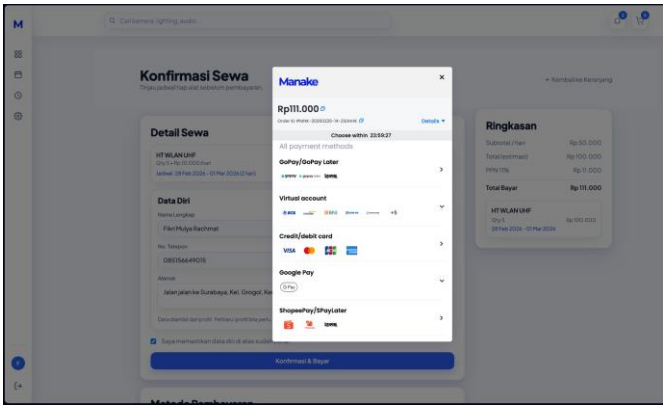

Figure 6. Midtrans Snap interface for payment method selection, including virtual accounts.

After payment, Midtrans sends a notification to the callback endpoint. The system validates the notification payload and verifies the SHA-512 signature using the combination of order_id, status_code, gross_amount, and the server key. Payment status is mapped as follows. It is marked as paid when capture or settlement occurs and fraud_status is not challenge. It is marked as expired when expire occurs. It is marked as failed when cancel, deny, or failure occurs. It is marked as refunded when refund or chargeback occurs. In addition, the system prevents duplicate webhook events by recording an event key in the payment_webhook_events table to support idempotency. [4], [6]

Once an order is marked “Paid”, Manake automatically generates an invoice that can be viewed online, printed, shared, or downloaded as a PDF. The invoice captures essential metadata, including the order ID, issue date, and the exact timestamp of payment confirmation. It also provides item details such as the rental period,

International Journal of Computer and Information System (IJCIS)
Peer Reviewed – International Journal
Vol : Vol. 07, Issue 01, June 2026
e-ISSN : 2745-9659
https://ijcis.net/index.php/ijcis/index

quantity, and daily rate, enabling customers to verify their booking. A transparent cost breakdown is displayed, showing the subtotal, 11% VAT, and the final total paid, for record keeping and audits. To support smooth handover, the invoice includes pickup instructions and a receipt reference for on-site verification. During collection, customers present the receipt number or order ID to staff. Figure 7 illustrates this invoice page and the available actions.

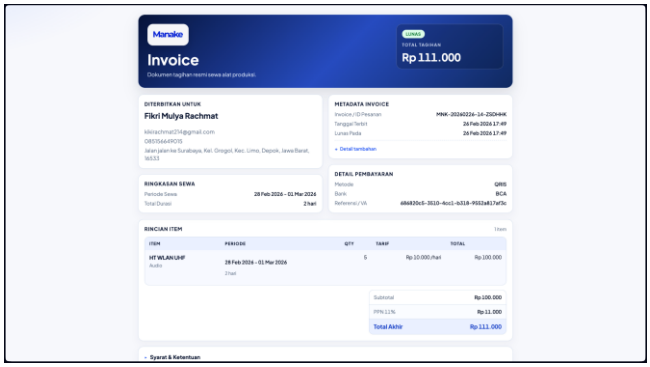

Figure 7. Invoice view with options to download PDF, print, and share.

3.5 Operational Admin Module and Monitoring

The admin panel serves as a central monitoring hub for daily operations. The dashboard presents key indicators, including incoming funds, rental revenue, estimated tax collected, and recorded damage fees, so admins can review performance at a glance. The order list supports filtering by payment status and rental status to prioritize tasks and reduce missed follow-ups. Through the equipment module, admins update stock levels, set readiness states such as ready, under maintenance, or unavailable, and adjust daily prices when required. A built-in website text editor enables updates to landing-page content without redeploying the application. As shown in Figure 8, these capabilities shift work from reactive checking of chats and transfers to proactive queue monitoring, unit preparation, and penalty handling when necessary.

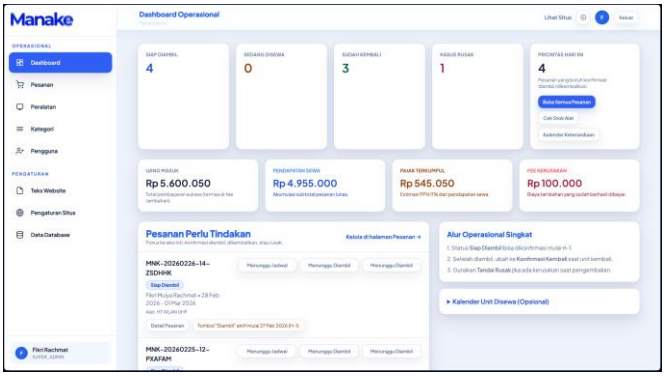

Figure 8. Operational admin dashboard and the list of orders requiring action.

On the admin side, the operational panel provides summary indicators such as incoming funds, rental revenue, estimated collected taxes, and damage fees. Admins can monitor orders, apply filters, view details, update payment and rental statuses, and add additional charges, for example damage penalties per item. As shown in Figure 9, each status or fee update can include an admin note and be sent as a notification to the user, making operational communication more structured and traceable.

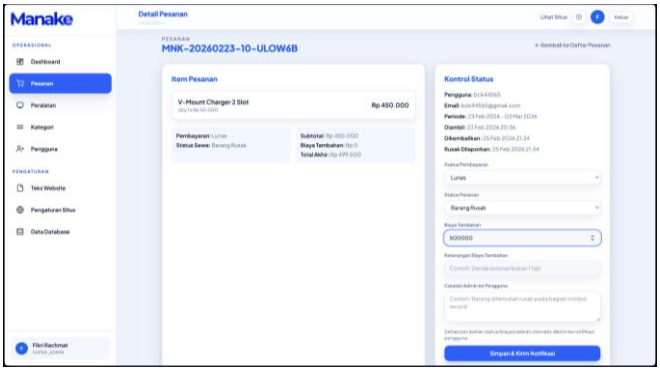

Figure 9. Order status control and additional fee or penalty input in the admin panel.

3.6 Penalty Policy, Additional Charges, and Financial Impact

Manake separates two types of additional charges. The first is a late return penalty, which is charged per order as a percentage of the daily rental cost. The second is a damage fee, which is charged per item with a minimum of 50% per item based on the admin's assessment. This separation is economically important. Late returns affect the opportunity cost of the entire order because the schedule remains blocked, while damage affects the cost of each individual asset unit. With structured additional fee recording and status logs,

the system improves accountability and reduces disputes because all changes are documented.

From a monetization and business scalability perspective, the system enables options such as dynamic pricing based on calendar demand, bundled packages such as camera plus audio plus lighting, membership programs for discounts or priority availability, and deposits that can be automated through the payment gateway. Over the long term, transaction and asset utilization data can support investment decisions, such as increasing stock for high-utilization items, and reducing inefficient asset purchases. MSME digitalization literature also notes that process digitalization improves decision quality and operational efficiency. [14]

The main risks in payment integration relate to callback reliability and transaction security. To address this, the system applies signature validation, event logging, and explicit status mapping. On the operational side, another risk is incorrect date or quantity input. This is mitigated through overlap validation and stock-based schedule locking. Black-box testing is selected to confirm that functions meet specifications without implementation bias. State-transition testing in other studies has been shown to be effective for systems with frequent status changes. [10], [11], [15]

From a financial perspective, payment gateway integration and automatic invoicing improve cash flow traceability and reduce payment confirmation waiting time. The additional fee mechanism for damage that can be billed through the system also increases the potential for asset cost recovery compared to manual tracking. From an operational risk perspective, signature validation and webhook idempotency reduce the risk of status manipulation and inconsistencies caused by duplicate callbacks.

Compared to rental practices that still rely on chat and manual transfers, the Manake system differentiates itself by enforcing operational policies as consistent system rules. These policies include an operational buffer, limited rescheduling, late-return penalties charged per order, and damage fees charged per item. Compared to Midtrans integration studies in other domains, Manake emphasizes the linkage between payment confirmation, schedule locking, and unit readiness as the core of the rental business process. [4], [5]

3.7 Business Impact and Scalability Discussion

Although this study focuses on system engineering, business impact should be discussed because the application is used in real operations. Digitalizing booking and payment helps speed up transaction confirmation, improves traceability, and reduces admin workload. As further development opportunities, a rental business can add bundled packages and membership programs for repeat customers and adjust pricing based on calendar occupancy levels.

From a cost perspective, using a payment gateway reduces manual payment verification time and lowers the risk of recording errors. Applying late return penalties per order and damage fees per item also serves as an asset protection mechanism. From an infrastructure perspective, using a managed database such as Supabase PostgreSQL allows capacity to scale with transaction growth without a large upfront server investment.

For technical scalability, increasing transaction load can be handled gradually through queue-based processing for webhooks and notifications, query optimization for availability checks by adding indexes on date fields and equipment_id, and separating access control for admin and user modules. This approach allows the system to grow without drastic architectural changes.

3.8 Research Limitations

The limitation of this study is that the evaluation is limited to black-box functional testing, without involving respondents for usability testing or user satisfaction measurement. In addition, the financial figures displayed on the dashboard are used as evidence of the system's aggregation capability rather than as audited financial reports. Future work can add performance testing such as load testing and, as summarized in Table 1, use the standardized payment and rental status definitions to support more precise measurement of booking processing time before and after implementation.

Table 1. Payment and Rental Status Definitions

Status	Operational Meaning
Pending Payment	The order is created, but payment has not been confirmed by Midtrans.
Paid	Payment is confirmed (capture or settlement), and the order can be processed.
Expired	Payment has passed the time limit (expire), and the order is not continued.
Failed	The transaction is cancelled or rejected (cancel, deny, failure).
Ready for Pickup	The admin marks the order as ready for pickup on H-1 based on the policy.
Item Picked Up	The renter has picked up the unit, and rescheduling is no longer available.
Item Returned	The unit has been returned and can be checked for condition and completeness.
Completed	The order is closed. Additional charges, if late return or damage occurs, will be added by the admin.

IV. BLACKBOX TESTING

Testing was conducted using the black-box method to verify that the main functions meet the specification, without examining the internal code structure. Test scenarios were designed to follow the business flow, including equipment and date selection, availability validation (including the one-day buffer), cart and cost calculation, payment via Midtrans Snap, status updates through webhooks, invoice issuance, limited rescheduling, and operational control by the admin. Testing was executed in a development environment (macOS) using PHP 8.5.2 and Laravel 12.49.0, with a PostgreSQL database on Supabase and the Midtrans environment. As summarized in Table 2, all black-box test scenarios were marked Valid, indicating that the system's core functions operated according to the specified business flow.

Table 2. Black-Box Test Scenario Summary

ID	Feature	Expected Result	Result
BB-01	Login / Registration	Users can log in or register successfully, and the session is stored.	Valid
BB-02	Catalog and Equipment Details	The catalog is displayed, and the equipment detail page shows stock and a valid blocked schedule.	Valid
BB-03	Availability Validation	The system rejects conflicting or overlapping dates and applies a one-day buffer.	Valid
BB-04	Add to Cart	Items are added to the cart with the selected dates, quantity, and duration.	Valid
BB-05	Cost Calculation	Subtotal is calculated, 11% VAT is added, and the final total is correct.	Valid
BB-06	Checkout Generates Snap	Checkout generates a Snap token or page and displays payment	Valid

		options.	
BB-07	Virtual Account Payment	A virtual account is generated and the transaction can be simulated to success.	Valid
BB-08	Valid Webhook Signature	A callback with a valid signature updates the payment status.	Valid
BB-09	Invalid Webhook Signature	A callback with an invalid signature is rejected and the status does not change.	Valid
BB-10	Payment Status Mapping	expire → Expired; cancel/deny → Failed; refund → Refunded.	Valid
BB-11	Webhook Idempotency	Duplicate callbacks do not trigger duplicate updates.	Valid
BB-12	Invoice After Payment	After status is Paid, the invoice can be viewed, printed, shared, and downloaded as a PDF.	Valid
BB-13	One-Time Reschedule	Rescheduling works once before pickup; duration and quantity remain the same; no refund.	Valid
BB-14	Reschedule Beyond Limit	A second reschedule attempt is rejected and the policy message is shown.	Valid
BB-15	Admin Status and Fees	Admin can update rental status, add damage fees (per item), and send notifications.	Valid
BB-16	Admin Late Return Penalty	Admin can add a late-return penalty (per order) and the charge is recorded.	Valid
BB-17	Additional Charges via Midtrans	If additional charges exist, the system creates an additional payment transaction and updates the status.	Valid
BB-18	Download Invoice PDF	The invoice PDF file downloads successfully and matches the order metadata.	Valid
BB-19	Admin Filter and Search	Admin can search orders or equipment and filter by status with correct results.	Valid

V. CONCLUSION

This study produces a web-based media equipment rental management system that integrates Midtrans Snap and webhooks to automatically synchronize payment status. The system enforces key operational rules, including a one-day buffer, a reschedule policy limited to one time before pickup with no refund, late-return penalties charged per order, and damage fees charged per item. Black-box testing on core scenarios shows that the main functions operate according to specification, including signature

validation and webhook idempotency mechanisms that support reliable transaction status updates.

For future development, the system can be enhanced through asset utilization analytics, rental package recommendations, real-time notification integration such as push notifications or the WhatsApp API, and the provision of a REST API to support integration with mobile applications and partner channels.

REFERENCES

[1] C. Mawardi, M. D. Rizaldi, and F. A. Syahputra, "Rancang Bangun Sistem Informasi Web to Print Terintegrasi Berbasis ERP (ODOO 13)," Jurnal Ilmiah KOMPUTASI, vol. 21, no. 2, pp. 169–176, 2022.

[2] APJII, "Jumlah Pengguna Internet Indonesia Tembus 221 Juta Orang," 2024. [Online]. Available: https://apjii.or.id/berita/d/apjii-jumlah-pengguna-internet-indonesia-tembus-221-juta-orang. Accessed: Feb. 28, 2026.

[3] Bank Indonesia, "QRIS Mencapai 57 Juta Pengguna dan 39,3 Juta Merchant (Semester I 2025)," 2025. [Online]. Available: https://www.bi.go.id/id/publikasi/ruang-media/news-release/Pages/sp_2717025.aspx. Accessed: Feb. 28, 2026.

[4] W. Djuwitaningrum and M. D. Hakim, "Pengembangan Sistem Reservasi Online Rental Graha Mustika Hall dengan Integrasi Payment Gateway," Jurnal Profita, vol. 6, no. 2, pp. 17–27, 2024.

[5] M. Hafizh, A. D. Cahyono, and I. Saputro, "Implementasi Payment Gateway Midtrans pada Sistem Informasi Pulsa Berbasis Web," Journal of Information and Communication Technology, vol. 1, no. 2, pp. 1–7, 2024.

[6] D. Permana and R. A. Pratama, "Integrasi Midtrans pada Aplikasi E-Commerce Berbasis Laravel," Journal of Informatic, vol. 4, no. 1, pp. 1–10, 2024.

[7] I. Owen and I. Fitrisia, "Heavy Equipment Rental Information System Development using the Prototyping Method," Journal of Applied Informatics and Computing, vol. 9, no. 2, pp. 475–482, Mar. 2025.

[8] A. Chalim, K. Surendro, and B. J. Kaleb, "Pengembangan Sistem Informasi Rental Peralatan Elektronik Berbasis Web," Jurnal Sistem Informasi Bisnis, vol. 6, no. 2, pp. 302–313, 2025.

[9] A. Manday and D. Indrawati, "Sistem Informasi Akuntansi Penyewaan Alat Berat Menggunakan Metode Waterfall," Journal of Applied Informatics and Computing, vol. 9, no. 1, pp. 245–252, Jan. 2025.

[10] N. Azkia, "State Transition Method Analysis for Testing the Interactive Rental Scheduling System," in Proc. 2024 IEEE Int. Conf. on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), 2024, pp. 1–5, doi: https://doi.org/10.1109/I4C62259.2024.10915158.

[11] A. Salim and R. Hartono, "Admin and Customer Ordering Information System Using Black Box Testing," in Proc. 2nd Int. Conf. on Software Engineering and Information Systems (ICSEIS), 2024, pp. 1–5, doi: https://doi.org/10.1145/3688968.3688991.

[12] S. Nurhayati and H. Saputra, "Penerapan Payment Gateway Midtrans pada Sistem Informasi Penjualan Berbasis Web," Jurnal Teknologi Informasi, vol. 3, no. 2, pp. 66–70, 2025.

[13] S. A. Buwono and R. Oktavia, "Digitalisasi UMKM melalui Sistem Informasi dan Pembayaran Digital," Journal of Computer and Information Systems Ampera, vol. 5, no. 1, pp. 81–92, 2024.

[14] D. Domański, "Digitalization and Business Model Innovation for SMEs: A Review," Journal of Cybersecurity and Privacy, vol. 3, no. 4, pp. 775–793, 2023, doi: https://doi.org/10.3390/jcp3040038.

[15] D. Ly, G. P. L. Overeem, R. Brinkkemper, and F. Dalpiaz, "The Power of Words in Agile vs. Waterfall Development," Journal of Systems and Software, vol. 219, Art. no. 112243, 2025, doi: https://doi.org/10.1016/j.jss.2024.112243.