

Implementation of Web-Based POS and Operational Laundry System at Laundry Melati

Muhammad Ainal Fikri Lubis¹, Eva Rahmawati^{2*}, Sri Diantika³

^{1,2,3} Faculty of Engineering and Informatics, Universitas Bina Sarana Informatika
ainalfikrilubis@gmail.com¹, eva.ehw@bsi.ac.id^{2*}, sri.szd@bsi.ac.id³

Abstract

The operational management at Laundry Melati currently faces several challenges, including manual transaction recording, vulnerability to human errors, delayed customer notifications, and unintegrated digital payment verification. This research aims to design and develop a web-based laundry operational and Point of Sale (POS) information system to optimize these business processes. The system was developed leveraging the Laravel 12 framework and MySQL for database management. Furthermore, external API gateways were integrated, including Midtrans for automated digital payment processing and WhatsApp gateway for real-time customer notifications. System architecture was modeled through Unified Modeling Language (UML) diagrams, including Use Case, Activity, and Class diagrams, alongside Entity Relationship Diagrams (ERD) and Logical Record Structures (LRS) to ensure a robust multi-role access control for Admin and Cashier roles. The system underwent evaluation using Black-box testing methods. The results demonstrate that all functional components, including automated transaction billing, real-time status updates, and digital payment callbacks, operate successfully with zero critical defects. In conclusion, the implemented system effectively enhances data validity, minimizes transaction processing time, and provides precise automated financial reporting.

Keywords: Access control, Information system, Laravel 12, Laundry management, Payment gateway

1. Introduction

The massive development of information technology has transformed operational paradigms across various business sectors, including Micro, Small, and Medium Enterprises (MSMEs). Amidst the increasingly competitive laundry service industry, the implementation of information systems has shifted from being merely a supporting asset to a primary necessity to maintain data accuracy, stability, optimization, and operational efficiency. Digitalization of operational management allows business owners to maintain full control over each business process, from garment intake to customer return. Consequently, utilizing a web-based Point of Sale (POS) system can facilitate business operations for MSMEs by managing sales data and reporting, as well as accelerating customer transactions to improve service efficiency [1].

However, in reality, many MSMEs, including small-to-medium-scale laundry operations such as Laundry Melati, still rely on traditional management methods. Based on direct observations and interviews with the business owner, it is evident that all logging activities are managed via physical ledgers and paper notes. Operating under manual recording methods introduces significant vulnerabilities to human errors, such as miscalculating total billing amounts based on clothing weights (kilograms) or individual pieces, losing or damaging physical receipts, and experiencing difficulties when retrieving historical transaction records. This condition is further exacerbated by the heavy administrative workload, where staff members must execute redundant data recapitulations daily, thereby consuming a significant portion of operational time.

The most crucial problem encountered by the owner of Laundry Melati is the inability of the manual recording system to provide accurate daily financial information. The business owner frequently faces difficulties in determining the daily Break Even Point (BEP). Without an automated calculation mechanism that links incoming cash, daily expenditures, and outstanding uncollected accounts receivable, the owner cannot accurately assess whether the daily operational activities have reached profit targets or incurred losses. This lack of real-time cash flow visibility significantly hinders the owner from making strategic decisions regarding future business development.

Beyond financial aspects, managing customer loyalty also presents a serious challenge. Currently, Laundry Melati lacks a database capable of automatically tracking the accumulated weight of laundry per customer. Despite this limitation, the owner intends to implement a customer appreciation program that provides bonuses or discounts to clients who achieve specific laundry weight thresholds. The absence of a systematic tracking framework causes the allocation of bonuses to rely solely on the owner's subjective memory, which inevitably lacks accuracy and scalability as the customer base expands.

Communication workflows and order status dissemination are also considered suboptimal. At present, customers must wait or inquire directly to ascertain whether their garments have been fully processed or remain in the washing queue. Digital receipts are also unavailable, forcing administrators to send manual chat alerts via WhatsApp to each customer individually. This operational bottleneck not only delays

the internal workflow but also compromises the perceived professionalism of the service. One of the core features of WhatsApp is notification delivery, which can seamlessly communicate details such as washing status, estimated completion times, and pickup notifications, thereby enhancing customer convenience by delivering direct alerts to their personal devices [2].

Therefore, this research aims to design and implement a web-based POS and operational information system at Laundry Melati. This system is intended to ensure that transaction data management, BEP calculations, and customer loyalty tracking can operate efficiently, in an integrated and responsive manner, while simultaneously supporting faster and more precise strategic decision-making amidst corporate operational dynamics.

2. Theoretical Basis

2.1. Definition of System

A system is a collection of interconnected or integrated elements designed to achieve a specific objective [3]. These components include inputs, processing, outputs, control mechanisms, and feedback that interact with the surrounding environment [3]. Furthermore, a system represents an assembly of interrelated parts that form a functional unity [3]. In essence, a system requires seamless collaboration among all its internal components to ensure that software functionalities operate successfully according to targeted goals.

2.2. Definition of Information

Information is defined as data that has been processed to increase the knowledge and understanding of its recipient [4]. This transformative process converts raw data bundles into highly useful assets with distinct practical values [4]. Additionally, information represents the outcome of data manipulation rendered in a meaningful form that depicts real-world facts to support decision-making processes [4]. It can be concluded that information within a laundry enterprise context is structured data derived from operational transactions presented clearly to facilitate strategic business adjustments.

2.3. Definition of Information Systems

An information system is a specialized workflow framework that coordinates human actors and computer systems to transform inputs into informational outputs to satisfy specific organizational objectives [4]. It combines computing technology, workforce roles, and strict procedural methods to handle, store, analyze, and disseminate data bundles systematically [3]. In essence, a web-based laundry information system serves to bridge raw transactional entries with web architecture, providing business owners with centralized monitoring capabilities over financial and operational variables.

2.4. Definition of Website

A website comprises a collection of interconnected web pages and digital files bound under a specific domain or subdomain layout hosted on the World Wide Web [5]. These pages are utilized to display multi-format information elements, including text, static images, animations, video dynamics, or audio configurations linked together through a structural network [5]. It can be concluded that dynamic web architectures are essential for service applications to enable flexible transactional record updates and provide real-time status updates directly to users.

2.5. Definition of Point of Sale (POS) and Operational Laundry Systems

A Point of Sale (POS) and laundry operational system is a specialized digital infrastructure engineered to automate core operational lifecycles within clothing wash services [6]. Implementing web-based management solutions represents a tactical milestone that minimizes data logging errors commonly caused by legacy manual practices while simultaneously restructuring internal financial transparency [6]. In essence, an integrated POS and tracking application streamlines the entire business journey ensuring highly organized business management.

2.6. Definition of Database

A database is a systematically organized and structured digital data array designed to facilitate easy access, manipulation, and periodic updates [7]. Within software development architectures, it acts as a centralized repository to store raw parameters before they are converted into valuable informative streams [7]. Moreover, database systems are defined by technical criteria, including data type specifications, structural entities, and validation boundaries to prevent storage anomalies [8]. It can be concluded that database setups are vital to house crucial application records securely.

2.7. Definition of PHP and Server-Side Scripting

Hypertext Preprocessor (PHP) is an open-source, high-level programming language centered on server-side scripting that allows developers to write custom business logic architectures [9]. Unlike client-side languages, PHP scripts are processed directly on the server host, which then returns standard web outputs back to the client perambulator interface [9]. In essence, PHP serves as the principal backend engine that executes transaction handling, pricing algorithms, and database queries securely.

2.8. Definition of Laravel Framework

Laravel is an open-source PHP framework utilized to design and deploy modern web applications quickly and effectively through highly expressive, streamlined syntax patterns [10]. It increases development speed significantly by supplying pre-built system tools, including

database migrations, the Artisan Command Line Interface (CLI), route controls, and the Eloquent Object-Relational Mapping (ORM) engine [10]. It can be concluded that the framework eliminates the requirement of composing complex raw SQL commands manually, facilitating rapid architecture implementation.

2.9. Definition of Frontend Integration (HTML, CSS, and Tailwind CSS)

Hypertext Markup Language (HTML) is the standard markup language used to build the foundational structural blueprint of web pages [11]. To enhance visual aesthetics, Cascading Style Sheets (CSS) collaborate with HTML to manage styles, color configurations, and responsive interface scales across varying device displays [11]. Furthermore, utility-first CSS frameworks like Tailwind CSS allow backend developers to arrange modern visual components efficiently without writing tedious custom style sheets from scratch [5]. In essence, the integration of structural markups and utility-first style layouts allows the generation of clean, professional interface panels for dashboards.

2.10. Definition of Interactivity Tools (JavaScript and Alpine.js)

JavaScript is a robust scripting language executed inside HTML documents to expand client-side interactive capabilities and capture user-triggered events smoothly [9]. To prevent the overhead of large frameworks, minimal libraries like Alpine.js are utilized to supply reactive attributes, such as data binding and event tracking, directly inside server-side templates [12]. It can be concluded that minimal reactive scripting engines enable real-time price calculations on cashier modules when clothes are weighed, eliminating unnecessary browser refresh actions.

2.11. Definition of MySQL Database

MySQL is a popular open-source Relational Database Management System (RDBMS) that processes, accesses, and modifies tabular data records via standard SQL directives [13]. The core engine relies on a database server that systematically intercepts instructions to create tables or log transaction streams, allocating them cleanly into storage directories [13]. In essence, MySQL provides a dependable relational base to manage authentication states, customer profiles, package tariffs, and digital callback variables securely.

2.12. Definition of Payment Gateway Integration

A payment gateway is an external transactional gateway architecture designed to provide versatile electronic payment channels for modern business operations [14]. Successful system connection demands proper API authentication using specialized credentials, including Server Keys and Client Keys, to secure transactional exchanges between platforms [14]. It can be concluded that implementing specialized payment gateway packages inside backend systems automates non-cash settlements like QRIS and Virtual Accounts with real-time status verification.

2.13. Definition of WhatsApp Gateway API

A WhatsApp Gateway is an automated messaging interface utilized to connect web platforms with cellular communication networks to automate text transmission workflows [15]. It facilitates real-time system alerts, such as automated invoice distribution, order queue state updates, and digital billing push notifications to client devices without manual inputs [15]. In essence, messaging gateway integrations optimize communication reliability, dramatically lowering administrative burdens while boosting operational professionalism.

2.14. Definition of System and Database Modeling (UML, ERD, and LRS)

The Unified Modeling Language (UML) supplies a standardized visual notation syntax to model object-oriented software dynamics and minimize conceptual communication gaps [16]. For storage architecture design, Entity Relationship Diagrams (ERD) trace structural associations, data fields, and explicit cardinality rules—such as one-to-one token binds or one-to-many customer transactions [17]. Concurrently, Logical Record Structures (LRS) act as a mapping blueprint to calculate the exact count of tables formed and handle foreign key placements during rapid system styling phases [10]. It can be concluded that standardized visual and database modeling provides a strict engineering blueprint to convert unstructured business workflows into organized object-oriented code before physical software programming begins.

3. Research Methods

This study implements a qualitative descriptive approach integrated with an adapted linear sequential Waterfall model configured specifically for a single-developer environment. Under this framework, the researcher operates independently as the system analyst, database architect, user interface designer, backend programmer, and system tester to engineer a web-based POS and operational platform.

3.1. Data Collection Methods

To obtain the necessary information and resolve the operational challenges at Laundry Melati, data collection was conducted through three distinct methods:

1. Interviews: Data collection was conducted through direct interviews with the business owner to identify administrative operational bottlenecks, financial reporting discrepancies, and notification delays
2. Observation: Data collection was conducted through direct observation of the daily laundry workflows, tracing the legacy paper-based ledger recording, and receipt handling mechanisms.
3. Literature Review: The research involved collecting data, frameworks, and theoretical constraints from various academic peer-reviewed journals and software engineering archives to support the system development phases [1, 2].

3.2. System Requirements Analysis

During the initial Waterfall lifecycle, operational vulnerabilities discovered during field analysis were mapped into six core functional software requirements dedicated to a single-user operator environment managed by the Owner:

1. Customer Data Management: A centralized ledger structure to register and search client credentials (names, WhatsApp contact strings, and home addresses).
2. Service Package Management: A dynamic index configuration to govern predefined laundry variations, such as express or standard weight-based tariffs.
3. POS Transaction Processing: A digital cashier platform engineered to process incoming customer orders and execute automatic transaction calculation parameters.
4. Payment Gateway Integration: Automated electronic settlement logging utilizing external banking interfaces to handle cashless interactions.
5. WhatsApp Notification Automation: Real-time communication workflows engineered to push digital invoices and washing states directly to customer cellular devices without manual text typing intervention [2].
6. Financial Reporting and Expense Tracking: An automated ledger utility designed to track revenue updates alongside cash outflows, supporting direct data extraction into multi-format reports.

3.3. System and Software Design

Locked functional criteria were transformed into rigid technical implementation blueprints before any programming blocks were composed:

1. Database Modeling (ERD & LRS): Architectural structures for the laundry_db repository on MySQL were mapped using Entity Relationship Diagrams (ERD) and Logical Record Structures (LRS), defining relationships across physical data tables.
2. Architectural Modeling (UML): Object-oriented relationships within the backend framework were mapped using Unified Modeling Language (UML) Class Diagrams to structure logical paths between controller directories and data models.
3. User Interface Design: Visual layout wireframes were organized around utility-first design rules to ensure comfortable administrative operations, binding reactive layout features to database components.
4. Data Flow and API Interfaces: Connections governing core application operations, external payment triggers, and automated notification alerts were structured to prevent operational processing conflicts.

3.4. System Implementation

The software construction stage translated design blueprints into executable source code using the Laravel 12 framework and PHP 8.2 backend engines:

1. Frontend Layouting: Clean, responsive interfaces were compiled inside Blade template engines utilizing Tailwind CSS and Alpine.js to handle reactive navigation structures such as sidebar menus.
2. Form Automation: Cashier input fields (create.blade.php) were optimized with jQuery to calculate total prices dynamically based on scale weight weights, while Select2 libraries were embedded to enable rapid lookup queries across client records.
3. Asynchronous Processing: Asynchronous backend interactions were implemented using the Native Fetch API (AJAX) to handle background data filtering and trigger notification tasks without causing full browser reload routines.
4. Tunneling Configuration: Local server testing environments (localhost) were exposed to secure public web endpoints utilizing ngrok tunnels to receive real-time automated transaction confirmation callbacks from external payment engines.
5. Backend Engineering: Core business operations and cross-table relations were coded using Laravel Resource Controllers, routing handlers, and Eloquent models, while physical table states inside MySQL were built utilizing database migrations.

3.5. System Testing and Maintenance

Verification and sustainability operations were structured to ensure long-term operational reliability:

1. Black-Box Testing: Functional integrity was verified purely from a user perspective through form entry simulations to catch operational defects, validate automatic serial invoice keys (e.g., INV-20260405-0001AA), test flexible calendar filtering via Flatpickr, and confirm automatic completion estimations computed via Carbon utilities (addHours, addDays).
2. API Verification: Safe transactional calls were tested using sandbox testing environments to verify that incoming webhook notifications dynamically update transaction rows to a paid status, while invoice dispatch calls were audited for instant text transmission.
3. Financial Reporting Audit: Revenue aggregation rules and expense logs were audited to verify clean output formatting when downloading files via DomPDF or Excel spreadsheet instances.
4. System Maintenance: Long-term operational safety was established through explicit security configurations inside environment files (.env), automated weekly database record exports into .sql safe storage files, asynchronous payload testing under data scaling conditions, and constant inspection of external API endpoint behaviors.

4. System Analysis and Design

This section outlines the software engineering modeling designed to transition the manual workflows of Laundry Melati into an automated web architecture. The proposed behavioral model captures the interactions between the primary user and the core application modules.

4.1. Proposed Use Case Diagram

The proposed system analysis explains the processing procedures of the point of sale and operational data within the laundry management application.

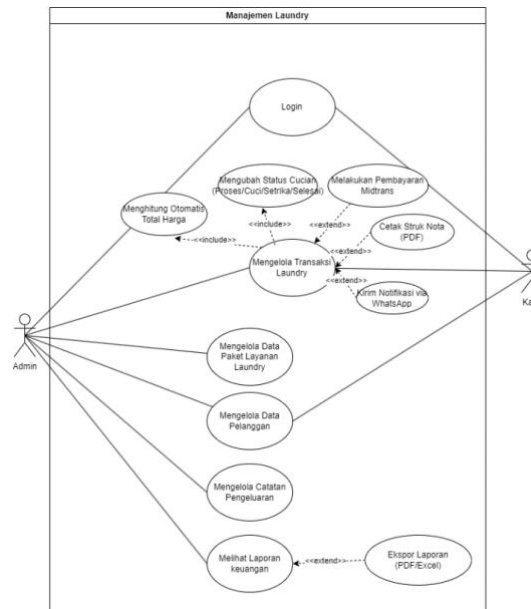


Fig 1: Proposed Use Case Diagram

4.2. Proposed Class Diagram

The structural code layout and database entity configurations engineered for the system are defined within the proposed class diagram parameters.

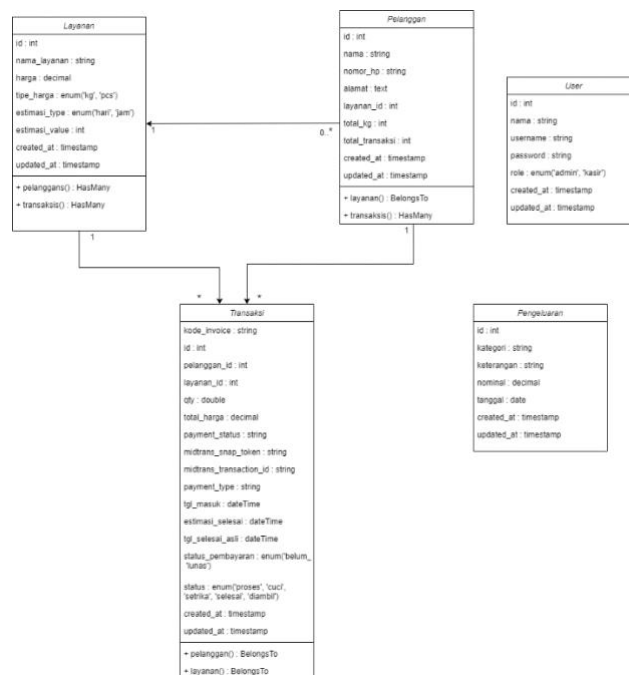


Fig 2: Proposed Class Diagram

5. Implementations and Testing

At the interface implementation stage, the researcher created the website display as follows:

5.1. Login Interface

The login interface serves as the primary gateway to secure the laundry operational panel. This page handles authentication routines via a customized server-side AuthController to prevent unauthorized access to backend modules.

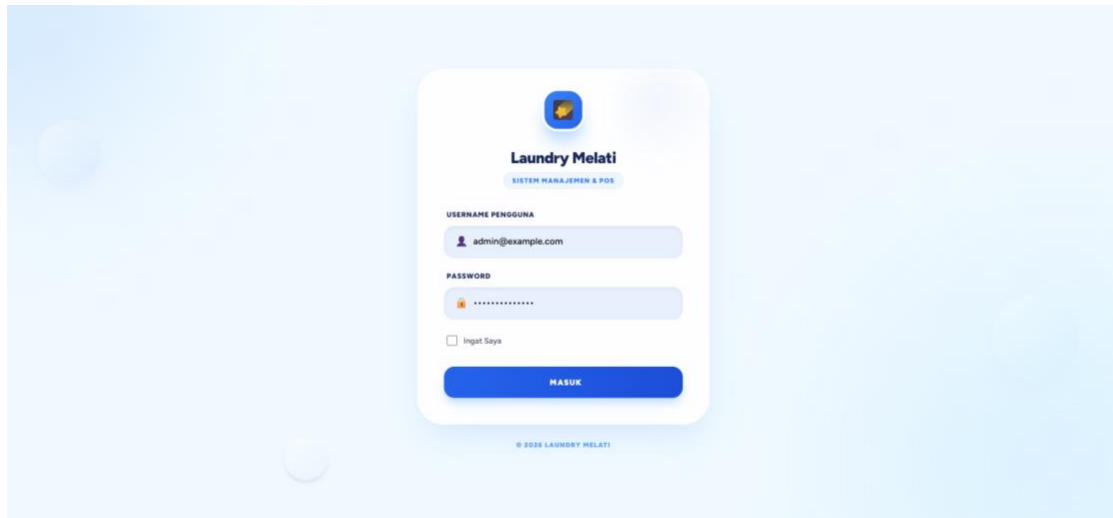


Fig 3: Login Interface

5.2. Dashboard Interface

The dashboard interface provides the business owner with real-time operational oversight. It displays quick visual summaries of daily transactions, incoming cash flows (income), and operational expenses (outcome) compiled dynamically using Tailwind CSS layout utilities and Alpine.js script components.

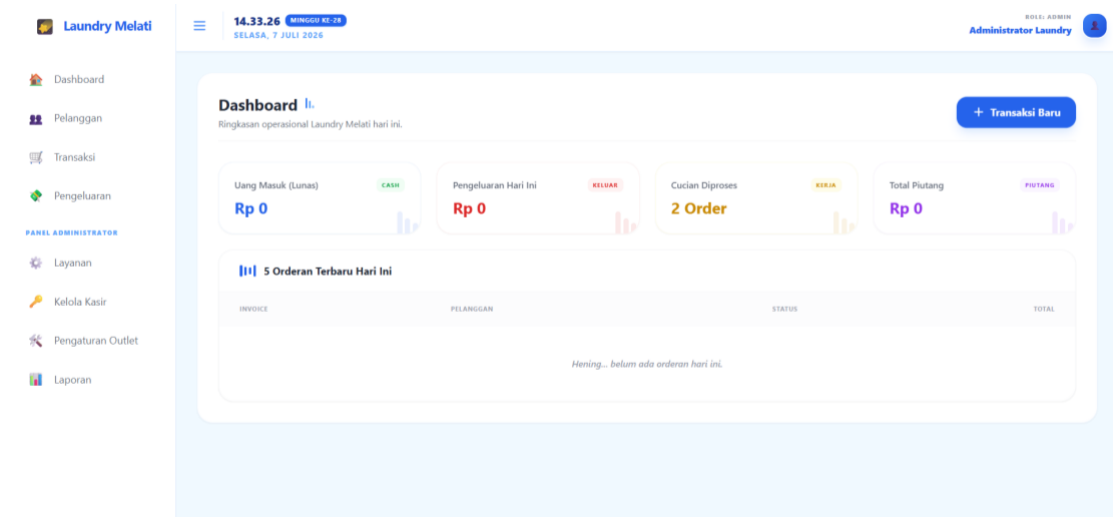


Fig 4: Dashboard Interface

5.3. Customer Management Interface

The customer management interface handles client data configurations linked directly to the pelanggan MySQL database table. This interface allows the operator to execute CRUD functions, search customer profiles, and track phone numbers for notification routing.

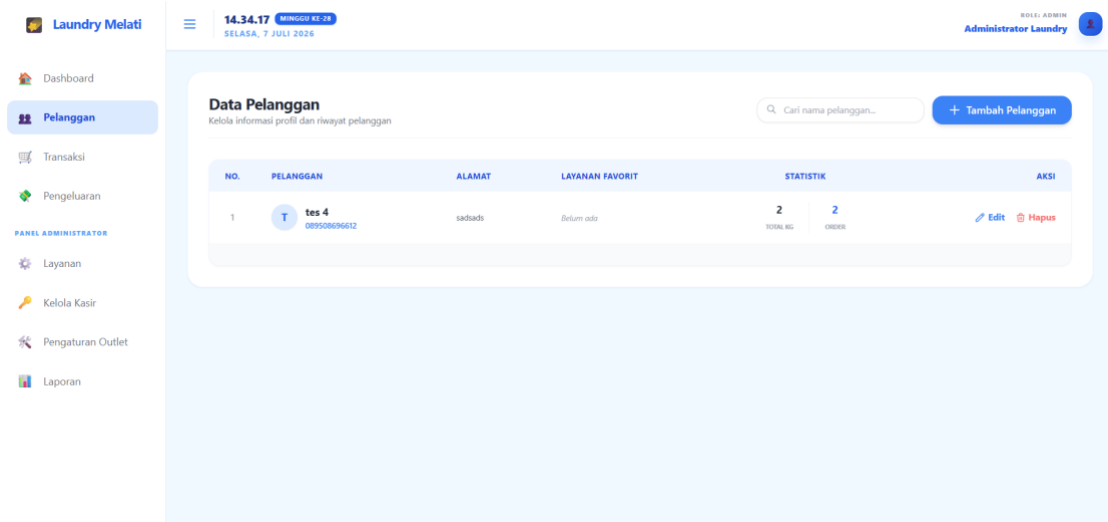


Fig 5: Customer Management Interface

5.4. POS Interface

The POS transaction interface represents the central core of the cashier module. This page utilizes Select2 drop-down controls for high-speed customer queries and integrates jQuery scripts to execute automated, real-time total price calculations based on clothes scaling weight inputs without requiring browser reloads.

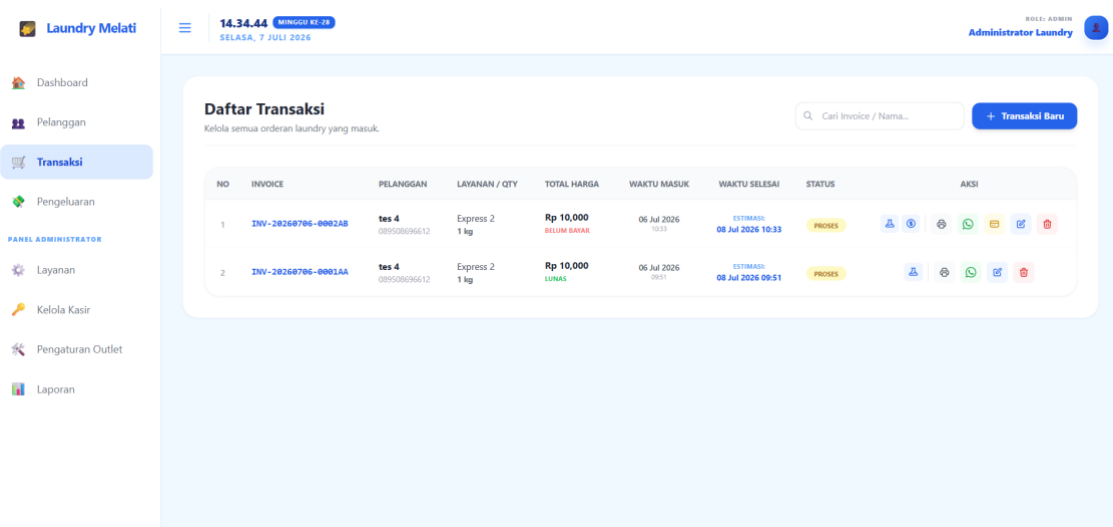


Fig 6: POS Interface

5.5. Midtrans Payment Gateway Interface

The payment gateway interface displays the integration of the Midtrans Snap modal embedded into the checkout workflow. This module intercepts sandbox banking network states, allowing cashless operations (QRIS and Virtual Accounts) to automatically trigger webhook callbacks that dynamically update transaction payment entries to a paid status.

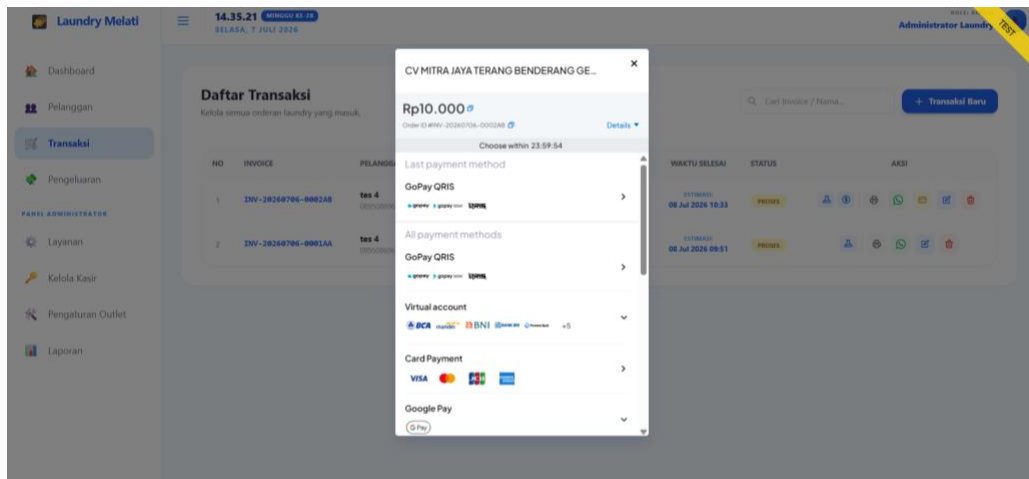


Fig 7: Midtrans Payment Gateway Interface

5.6. Expenses Tracking Interface

The expense tracking interface is a dedicated page designed to record and manage all operational expenditures of the business. This interface allows the user to log daily routine costs—such as laundry detergents, utility bills, and equipment maintenance—efficiently to maintain full visibility and transparency over all cash outflows.

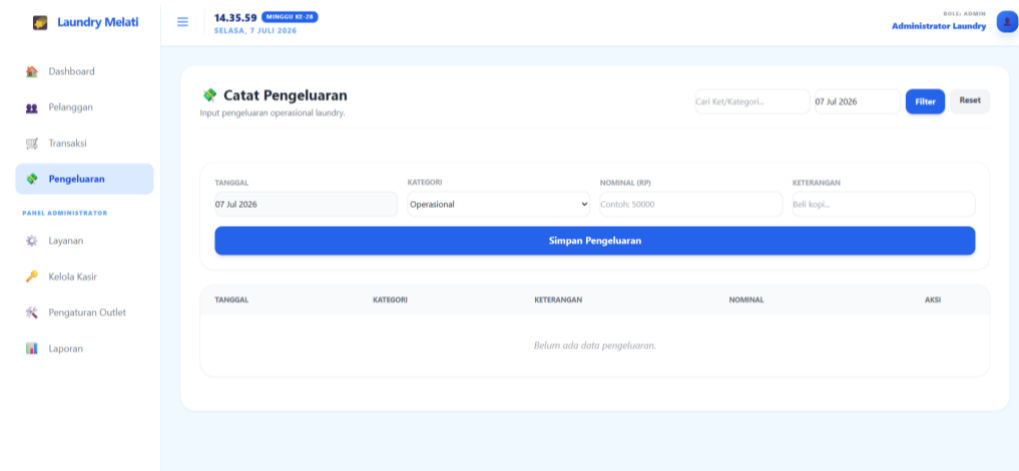


Fig 8: Expenses Tracking Interface

5.7. Financial Reporting Interface

The financial reporting interface summarizes data from both transaction revenues and recorded expenses to present a comprehensive financial balance. This interface supports strict chronological filtering options (daily, weekly, and monthly) and enables the owner to view dynamic profit calculations as well as download clear, print-ready financial reports and spreadsheets.

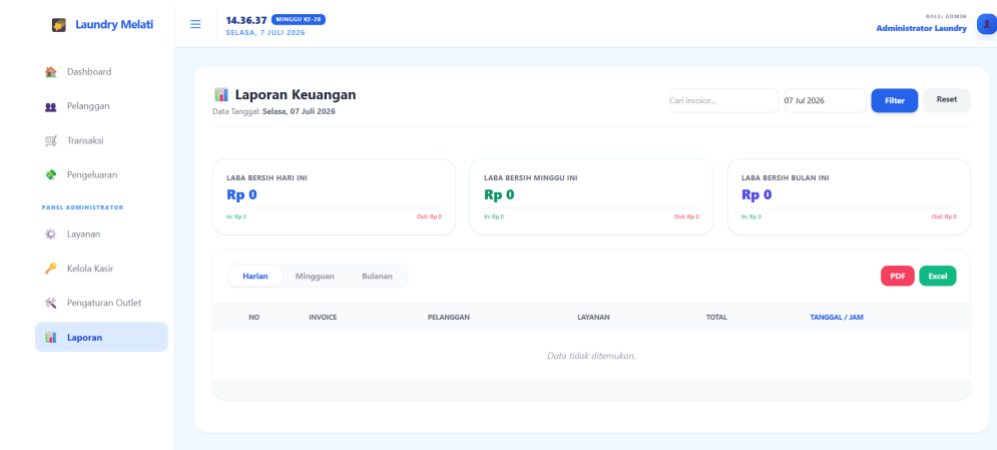


Fig 9: Financial Report Interface

5.8. Test Result Conclusion

Based on the results of previous tests conducted by the researcher, it can be concluded that the web-based Point of Sale (POS) and operational system at Laundry Melati that was designed and created can run and function well.

6. Conclusion

Based on the research conducted, several conclusions can be drawn:

1. The previous operational and transaction tracking system at Laundry Melati was still carried out manually, such as recording customer entries, weighing calculation records, and storing financial data in physical ledgers and paper notes.
2. The web-based Point of Sale (POS) and operational system that was designed and built has successfully simplified the process of processing transactions, automated weight-to-tariff calculations, tracking daily expenses, and managing financial reports at Laundry Melati compared to previous manual methods.
3. System trials showed that the system operates reliably and optimally, helping the owner accelerate operational workflows through automated features, such as real-time digital payment confirmations and automated message notifications sent directly to customer devices.

References

- [1] M. A. Nurullah, Y. Nyura, and F. Metandi, "Rancang bangun sistem informasi POS berbasis web laravel," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 4, no. 2, pp. 6424–6436, 2025.
- [2] M. Fauzia, A. Ginanjar, I. Supriadi, I. Abadi, and W. P. Sari, "Implementasi algoritma priority scheduling dan FIFO serta notifikasi WhatsApp pada sistem informasi manajemen laundry berbasis website," *Jurnal Teknologi Informasi Digital (JTID)*, vol. 1, no. 2, 2025.
- [3] Aidah, O. Arifudin, I. Kartika, and T. Ibrahim, "Pengembangan sistem informasi manajemen dalam dunia pendidikan," *Jurnal Tahsinia*, vol. 5, no. 6, pp. 966–977, 2024.
- [4] A. Zulfa, T. Ibrahim, and O. Arifudin, "Peran sistem informasi akademik berbasis web dalam upaya meningkatkan efektivitas dan efisiensi pengelolaan akademik di perguruan tinggi," *Jurnal Tahsinia*, vol. 6, no. 1, pp. 115–134, 2025.
- [5] P. Maharani, "Pengembangan website PT. Rantingin Digital Indonesia menggunakan framework Next Js dan Tailwind CSS," *Repeater: Publikasi Teknik Informatika dan Jaringan*, vol. 3, no. 1, pp. 129–137, 2025.
- [6] F. Herlambang, B. B. Pramudito, and T. H. Hasibuan, "Rancang bangun sistem informasi penatu (laundry) berbasis web," *Prodi Sistem Informasi ITB Swadharma*, Jakarta, 2025.
- [7] Rachman, I. D. Mumpuni, W. A. Dewa, et al., *Big Data dan Manajemen Basis Data Terdistribusi PT. MIFANDI MANDIRI DIGITAL*, Jakarta: PT. Mifandi Mandiri Digital, 2025.
- [8] P. E. Putri and K. Khairunnisa, "Perancangan data base sistem pembelajaran sekolah dasar menggunakan ERD," *TEKNOFILE: Jurnal Sistem Informasi*, vol. 3, no. 5, pp. 314–326, 2025.
- [9] C. Christian and A. Voutama, "Implementasi aplikasi antrian pencucian mobil berbasis web menggunakan PHP, JavaScript, HTML, CSS dan UML," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 2, pp. 201–210, 2024.
- [10] R. I. Melyani, Rosita, and S. Aji, "Pengembangan sistem informasi penggajian berbasis web menggunakan framework laravel dengan metode agile software development," *JASIKA (Jurnal Sistem Informasi Akuntansi)*, vol. 3, no. 1, pp. 31–36, 2023.
- [11] P. Sari, Azzahrah, I. F. Qathrunada, N. Lubis, and T. Anggraini, "Perancangan sistem absensi pegawai kantor secara online pada website berbasis HTML dan CSS," *Fakultas Ilmu Komputer dan Teknologi Informasi UMSU*, Medan, 2022.
- [12] C. Porzio. "Alpine.js Documentation." *alpinejs.dev*. <https://alpinejs.dev> (accessed Jul. 7, 2026).
- [13] R. A. Aziz, A. Sansprayada, and K. Mariskhana, "Perancangan sistem informasi manajemen cuti karyawan pada PT SuMoIn dengan menggunakan PHP dan MYSQL," *Jurnal Minfo Polgan*, vol. 14, no. 1, pp. 15–23, 2025.
- [14] Y. Fatman, N. K. Nafisah, and P. B. J. Pambudi, "Implementasi payment gateway dengan menggunakan midtrans pada website UMKM Geberco," *Jurnal KomtekInfo*, vol. 10, no. 2, pp. 64–72, 2023.
- [15] Dibah, C. Sanyoko, and R. Imani, "Pemanfaatan Fonnte dalam membangun sistem informasi pengiriman pesan berbasis website untuk optimalisasi komunikasi di Bakesbangpol," *Bakesbangpol*, Tech. Rep., 2025.
- [16] A. Binangkit, A. Voutama, and N. Heryana, "Pemanfaatan UML (Unified Modeling Language) dalam perencanaan sistem pengelolaan sewa alat musik berbasis website," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 7, no. 2, pp. 88–95, 2023.
- [17] R. Mukhlis and R. Santoso, "Perancangan basis data perpustakaan universitas menggunakan MySQL dengan physical data model dan entity relationship diagram," *JoTI (Journal of Technology and Informatics)*, vol. 4, no. 2, pp. 76–84, 2023.