J-INTECH (Journal of Information and Technology)
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia SK No. 10/C/C3/DT.05.00/2025
E-ISSN: 2580-720X || P-ISSN: 2303-1425

Designing a Web-Based Dropship Management System for PT Xarana Djaya Motor

Theresia Gabriella[1], Bhustomy Hakim[2*]
[1,2]Bunda Mulia University, Faculty Technology and Design, Information Systems, Jl. Jalur Sutera Barat Kav 7– 9 Alam Sutera Jakarta, Indonesia

Keywords
Dropship; Information System; E-Commerce; Laravel; UML; Inventory

Corresponding Author:
bhakim@bundamulia.ac.id

Abstract
Indonesia's online retail is expanding alongside the national digital economy: 56.1% of global internet users shop on weekends, with Indonesia ranking ninth at 59.3%. The digital economy is projected to reach IDR 5,800 trillion by 2030 (USD 360 billion), and in 2023 Indonesia accounted for 40% of ASEAN's e-commerce market. In this context, PT Xarana Djaya Motor, a multi-brand motorcycle spare-parts distributor, runs a dropshipping model with multiple online sellers. Yet orders and shipment confirmations are still coordinated via chat and reentered manually, causing duplicate entries, processing delays, and limited real-time stock visibility. This study designs and implements a web-based Dropship Management Application that centralizes catalog, order, shipment, and return workflows. Analysis uses UML/ERD; implementation employs Laravel (PHP) and MySQL. Alpha/black-box testing validates key rules (one shipment per order) with multiple items and partial returns at the order-item level. Results indicate reduced manual handling, improved data accuracy, and clearer status visibility as supported by centralized workflows and functional (black-box) acceptance tests. The work contributes a reusable reference architecture for SMEs, with future extensions to marketplace API integration, automated label generation, analytics dashboards, and mobile access.

1. Introduction

Indonesia's digital economy continues to expand and is accompanied by sustained growth in online retail activity. Recent statistics report that 56.1% of global internet users shop online on weekends, with Indonesia ranking ninth at 59.3%[1]. Projections further estimate the national digital economy will reach approximately IDR 5,800 trillion (USD 360 billion) by 2030, while in 2023 Indonesia accounted for 40% of ASEAN's e-commerce market share[2]. These indicators underscore a favorable context for firms to strengthen operational readiness and information systems supporting online sales at scale.

PT Xarana Djaya Motor is a distributor of multi-brand motorcycle spare parts that collaborates with multiple online sellers under a dropshipping arrangement, wherein sellers do not hold inventory and instead forward paid orders to the supplier for direct fulfillment to customers[3]. In practice, however, order details and shipment confirmations are exchanged via chat applications and reentered manually by an administrator, leading to duplicated entries, processing delays, and fragmented reporting across sellers. More broadly, many

SME dropship operations rely on a combination of chat, spreadsheets, and marketplace dashboards that are not designed for supplier–dropshipper coordination[4]. While these tools are convenient for daily communication or individual store operations, they exhibit critical shortcomings when used as the primary backbone for multi-seller fulfillment: (i) data is dispersed across messages, files, and separate seller accounts, making reconciliation time-consuming; (ii) stock information is not shared as a single real-time reference for all sellers, increasing the risk of overselling and stock-outs; (iii) order lifecycle tracking (order–shipment–return) is inconsistent because status updates are recorded in different formats; and (iv) consolidated reporting and billing across multiple dropshippers cannot be produced reliably without repeated manual recaps. This fragmentation prevents a single source of truth for stock, order status, returns, and consolidated billing, thereby increasing inconsistencies, rework, and delayed decision-making.

Prior work highlights the central role of consumer trust and service quality in dropshipping satisfaction[5]. Moreover, demand uncertainty and limited supplier transparency emphasize the need for real-time inventory visibility to inform selling decisions and prevent stock-outs[6]. Given these conditions, the company requires an integrated web-based information system that serves as a hub between the firm and its dropshippers. Core requirements include digitized order capture and status tracking, automatic stock updates, centralized product information, and structured logs of shipments and returns that are accessible to both administrators and sellers. Such a system is expected to reduce data entry errors, shorten processing time, and improve operational transparency, thereby enhancing seller experience and supporting the onboarding of new dropshippers.

To address the problem, this study designs and implements a dropship management application using a structured, sequential Waterfall process analysis, design, implementation, testing, and maintenance—as an appropriate choice for relatively stable requirements[7]. The solution is realized with PHP (Laravel) following the Model–View–Controller paradigm and a MySQL relational database to enforce consistency and maintainability in development[8]. The objectives are (i) to engineer a system aligned with PT Xarana Djaya Motor's operational needs and (ii) to evaluate it through alpha/black-box testing against specified business rules. The results aim to provide a practical reference for small-to-medium enterprises operating dropship-based e-commerce in Indonesia.

2. Research Method

This study adopts the SDLC Waterfall model as the sole development methodology to engineer the dropship management system for PT Xarana Djaya Motor. Waterfall proceeds sequentially through clearly delimited phases requirements analysis, system design, implementation, testing, and maintenance with formal reviews and deliverables at the end of each phase[9]. It is chosen for its structured control, document-driven handoffs, and suitability to relatively stable requirements, enabling traceability from needs to verified functionality[10].

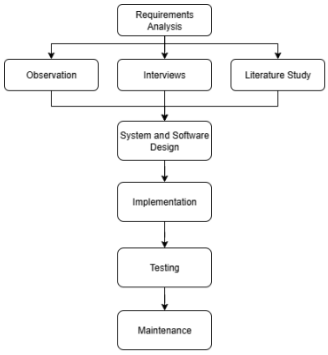

Figure 1. Research Method

The study began with a structured requirements analysis focusing on the comprehensive elicitation of system needs so that the resulting application would be maximally useful to its users[11]. Data were gathered through

a literature review of books, journals, and relevant scientific articles; direct observation at PT Xarana Djaya Motor to understand operational workflows and bottlenecks; and semi-structured interviews with key stakeholders (the commissioner and the operations manager). Analysis of the as-is process revealed several issues: vulnerability to recording errors and delays because orders were handled manually and were not integrated; the absence of a centralized repository for dropshipper, transaction, and sales-performance data, which hindered monitoring and evaluation; and manual reporting that often produced inconsistencies between stock figures and actual conditions, thereby slowing decision-making.

The system and software design phase translated these requirements into formal models. Business processes were modeled using Business Process Modeling Notation (BPMN) to capture the intended (to-be) operational flow[12]. Functional behavior and structure were specified with Unified Modeling Language (UML), including a use-case diagram to outline actor–system interactions, activity diagrams to detail control flows[13]. A normalized Entity–Relationship Diagram (ERD) defined the database schema supporting core entities and their relationships[14].

Implementation realized the design into a working web application. Development consisted of coding with the Laravel framework (PHP) alongside HTML, CSS, and Bootstrap, using a modern code editor (e.g., Visual Studio Code) and auxiliary tools as needed. Unit testing accompanied construction to verify each module against its specification and to ensure proper handling of inputs, validation rules, and data integrity[15].

Integration and system testing evaluated the end-to-end behavior of the integrated system to determine whether it met the specified functions or required corrective actions. Alpha testing was conducted using a black-box approach, deriving test cases directly from functional requirements and business rules to validate inputs, outputs, and state transitions without inspecting internal code[16]. Test outcomes were analyzed to confirm alignment with the previously defined requirements and design specifications, and any defects were addressed accordingly[17]. Consistent with a Waterfall-style SDLC, the project was executed through the testing phase; maintenance activities are planned as future work. In addition, this study proposes supplementary quantitative indicators to be collected during beta deployment.

3. Result and Discussions

Following the Waterfall process, the presentation proceeds from modeling outputs to implementation evidence and evaluation. The as-is business process is first analyzed with BPMN to identify bottlenecks caused by chat-based ordering, duplicated re-entry, and fragmented stock visibility across sellers[18]. hese findings align with prior observations that fragmented tools (chat/spreadsheets/marketplace dashboards) limit end-to-end coordination and weaken inventory transparency in distributed e-commerce operations, increasing rework and decision latency. Therefore, the proposed solution is formalized using UML use case, activity, and ERD to translate the identified gaps into enforceable system functions and a controlled data backbone.

Figure 2 summarizes the system scope through two roles—Admin and Dropshipper—clearly separating responsibilities and access boundaries. Beyond describing features, this model operationalizes process integration by ensuring that catalog control, order status transitions, return handling, and consolidated billing are managed within one supplier-governed workflow. This is significant because many dropship implementations and marketplace-native flows typically emphasize ordering and basic tracking, while returns and reconciliation remain handled externally (e.g., chats or spreadsheets), which reduces traceability and increases disputes. The include/extend relations represent conditional controls (e.g., proof-of-payment and reminders) consistent with use-case models as interaction contracts that support business rule enforcement and accountability[19].

Figure 2. Usecase Diagram Website Dropship Management

Figure 3 (Activity Diagram) explains the end-to-end operational flow from product onboarding to fulfillment, returns, and billing. Interpreting this model, the key contribution is the explicit visibility of decision points and parallel activities that previously occurred informally, enabling identification of operational risk areas (e.g., bulk upload validation, status update dependencies, and return approval gates). Such explicit control points strengthen traceability and standardization, which are commonly cited as prerequisites for consistent execution in integrated e-commerce processes[20].

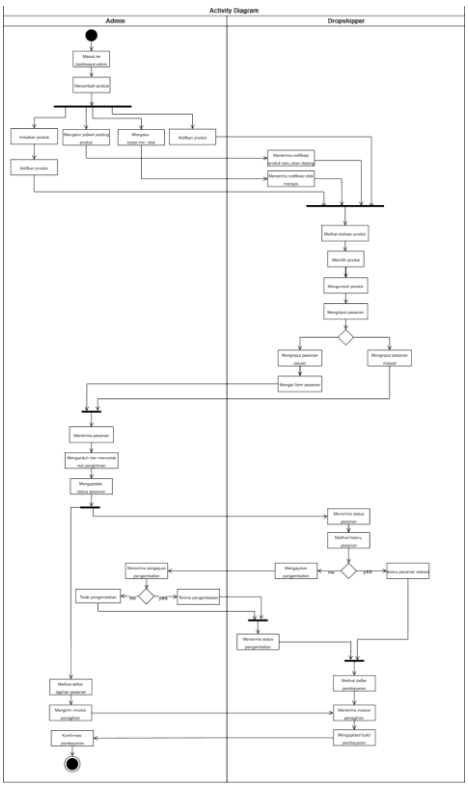

Figure 3. Activity Diargam Website Dropship Management

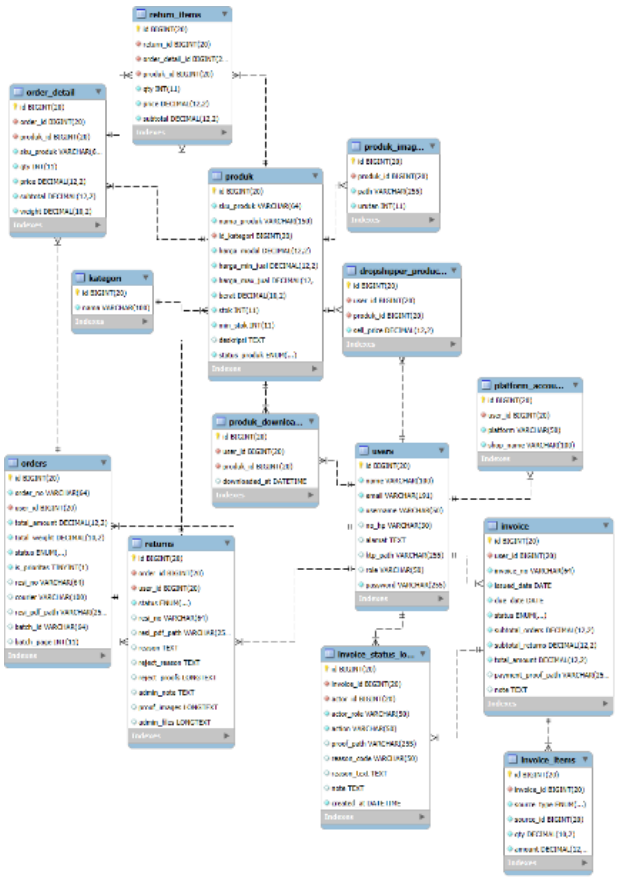

Figure 4. ERD Website Dropship Management

Figure 4 presents the ERD for an integrated dropship workflow, translating the end-to-end operational process into a normalized and auditable data model[21]. The design ties Dropshipper–Orders–Order Items to Products and Category, enforces one Shipment per order, and supports item-level partial returns. Importantly, billing (Tagihan and Detail_tagihan) and multi-marketplace management (Dropshipper_platform to Platform) are incorporated to enable consolidated reconciliation—an aspect frequently treated as separate or loosely controlled in earlier dropship schemas that focus on order/track only.

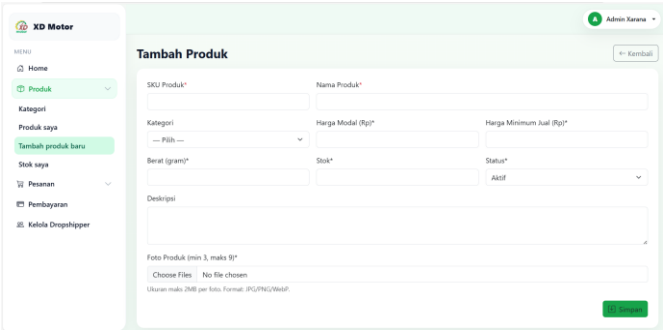

Figure 5. Page add product admin

Figure 5 (Admin – Add Product) illustrates how the system establishes a controlled "product master" by capturing canonical catalog attributes (e.g., SKU, name, category, cost, minimum selling price, weight, stock, status, and description) and validating product images through explicit constraints (format, size, and count). In the context of dropship operations that often depend on chat messages and spreadsheets for product updates,

this interface is significant because it standardizes catalog data at the supplier side, reducing ambiguity in item identity and listing details when the same products are marketed by multiple dropshippers.

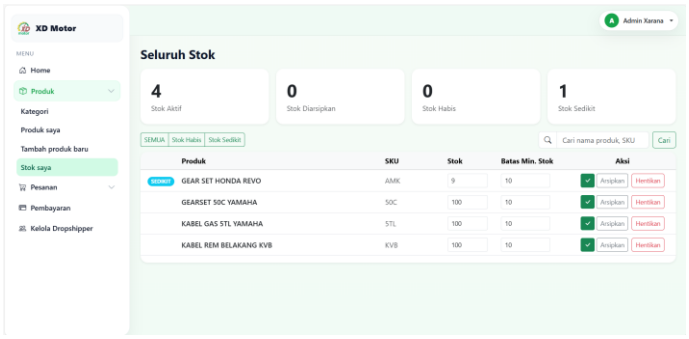

Figure 6. Page stok admin

Figure 6 presents the Admin Stock page, which summarizes inventory through four cards (active, archived, out-of-stock, and low-stock) and provides quick filters plus a search field by product name or SKU. The table lists items with SKU, current stock, and an editable minimum-stock threshold that triggers low-stock flags; chips/badges mark products below the threshold. Action controls enable archiving or re-enabling items, offering role-based control over catalog visibility. Edits are persisted to the inventory tables and reflected in dropshipper listings and exports, supporting timely monitoring and replenishment decisions.

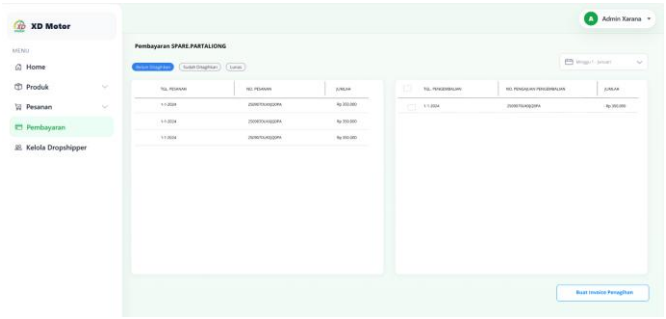

Figure 7. Page payment admin

Figure 7 depicts the Admin Payment page, which consolidates billable transactions for a selected dropshipper and period. Orders appear on the left and return adjustments on the right; negative amounts from approved returns offset the invoice total. Tabs (unbilled, invoiced, paid) track billing status, while checkboxes allow the administrator to include or exclude specific entries before generating an invoice. The “Create Billing Invoice” action compiles the selected items into an invoice record and updates lifecycle states for subsequent reconciliation.

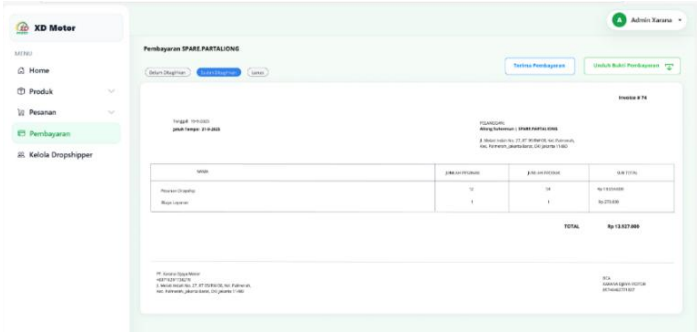

Figure 8. Page invoice

Figure 8 shows the Admin Billing–Invoice view for a selected dropshipper after an invoice has been issued. The interface renders the invoice inline with metadata (invoice number, issue date, due date, customer identity) and provides state tabs (unbilled, invoiced, paid) plus actions to acknowledge receipt of payment or download the payer's proof. The detail table aggregates billable components such as dropship orders and service fees with counts of orders/products, subtotals, and a computed grand total. Confirming payment transitions the invoice to paid, locks the associated line items, and records an audit entry for reconciliation and reporting.

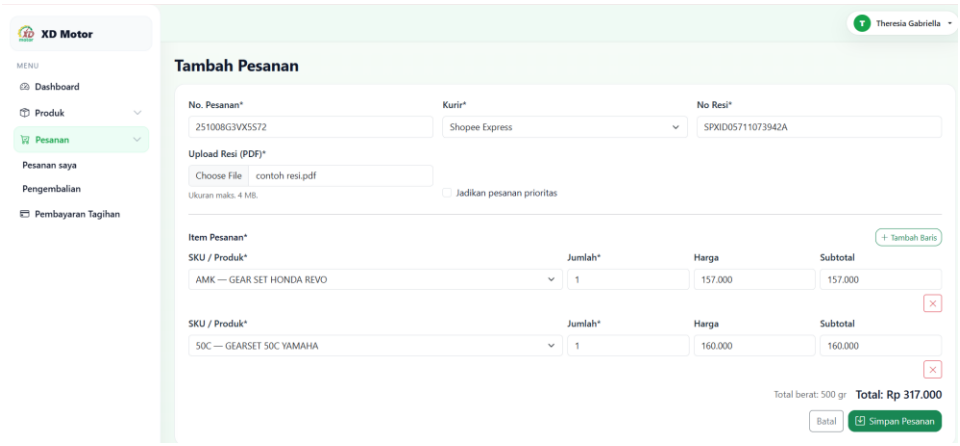

Figure 9. Page add order

Figure 9 shows the order form captures the external order number, courier, and marketplace tracking number, with an optional PDF upload of the receipt; a priority flag elevates urgent orders in downstream queues. Line-item rows bind SKUs to quantities and unit prices, compute subtotals and total weight, and validate required fields. Submission persists a single order header with multiple items, aligning with the rule of one shipment per order and preparing the record for fulfillment and billing.

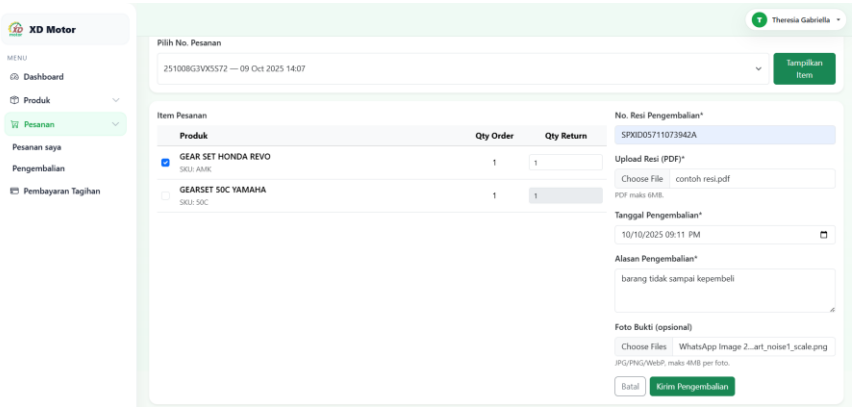

Figure 10. Page add return order

Figure 10 the return interface begins with order selection, then renders the associated items so the user can mark one or more lines and specify per-item return quantities, thereby enabling partial returns at the order-item level. Required fields include the return tracking number, return date–time, and a reason field; attachments accept a PDF receipt and optional photographic evidence. On submission, the system records a structured return request and updates order/ledger states for subsequent review and invoice adjustment.

Tabel 1. Blackbox Testing

No	Tested Page	Test Data	Expected Result	Output	Test Results	User Fedback/Usability
1	Admin – Add Product	Fill SKU, Name, Category, Cost, Minimum Price, Weight, Stock, Status, Description; upload 3–9 images ≤2 MB	Inputs validated; product saved; appears in catalog	Product created and listed	Valid	Clear form; required-field indicators help completion
2	Admin – Stock	Change minimum-stock threshold; use search and quick filters; archive/enable item	Counters (active/archived /out-of-stock/low-stock) and badges update; item visibility toggles	Inventory updated; flags refreshed	Valid	Fast controls and responsive search
3	Admin – Payment (Build Invoice)	Select dropshipper & period; tick orders and returns; click Create Billing Invoice	Invoice generated with selected entries; returns reduce total	Invoice record created	Valid	Separation of orders/returns eases selection
4	Admin – Billing / Invoice View	Review invoice; click Receive Payment; download proof	Status changes to Paid; line items locked; proof available	Invoice marked paid	Valid	Inline invoice layout is readable
5	Dropshipper – Product Display	Search by SKU/name; filter by category/status; select items; download	Filtered listing; catalog/export file downloaded	Filtered list & export file	Valid	Informative cards; badges aid scanning
6	Dropshipper – Add Order	Enter Order No., Courier, Tracking; add ≥2 items; upload PDF receipt	Order saved (one header, multiple items); totals and weight computed	Order created with totals	Valid	Auto-calculation improves accuracy

All scenarios executed as valid, with no high-severity defects. Critical rules single shipment per order, partial returns at the order-item level, low-stock thresholding, and invoice aggregation with return offsets behaved as specified. Usability notes were positive (clear forms, responsive filters, informative badges and totals), indicating that the implemented interfaces support accurate data entry and efficient navigation. These outcomes satisfy the acceptance criteria for the alpha phase and provide a reliable basis for subsequent deployment and/or broader user testing.

Tabel 2. Supplementary quantitative indicators

Indicator	How to measure	Baseline (Before) [B]	With System (After) [A]	Calculation	
Manual entries per	Count typed fields required end-to-end, including re-	B1 = 44.2	A1 = 18.1	Reduction = (B1 – A1)/B1 × 100%	(44.2–18.1)/44.2 = 59.1% fewer entries

Indicator	How to measure	Baseline (Before) [B]	With System (After) [A]	Calculation	
order (fields typed)	entry across chat/Excel/system				
Processing time per order (minutes)	Stopwatch time from order submission to 'Perlu Proses' to 'dikirim' status	B2 = 21.4 min	A2 = 11.3 min	Reduction = (B2 – A2)/B2 × 100%	(21.4–11.3)/21.4 = 47.2% faster

Table 2 shows clear efficiency gains after the system was implemented. Manual entries per order decreased from 44.2 (B1) to 18.1 (A1), a 59.1% reduction, indicating less duplicate typing across tools. Processing time per order dropped from 21.4 minutes (B2) to 11.3 minutes (A2), a 47.2% improvement, reflecting faster status progression to "dikirim."

Combined with the black-box testing results, these findings confirm that the system not only functions correctly according to the test scenarios, but also delivers measurable operational improvements by reducing manual work and accelerating order processing.

4. Conclusions and Future Works

This study engineered and validated a web-based dropship management system for PT Xarana Djaya Motor using a Waterfall process and a model-driven specification (BPMN, UML, ERD) implemented on Laravel–MySQL. Alpha black-box testing and Supplementary quantitative indicators across administrator and dropshipper workflows met the acceptance criteria single shipment per order, partial returns at the order-item level, inventory thresholding, and invoice aggregation with return offsets and the results indicate reduced manual handling, improved data accuracy, and clearer status visibility along the order–shipment–return lifecycle. The resulting specification and data model constitute a reusable reference architecture for SMEs operating dropship-based commerce, though the present scope excludes automated payment integration, direct marketplace APIs, and post-deployment maintenance beyond the alpha window.

Future work should prioritize marketplace API integration (e.g., Shopee, TikTok Shop, Tokopedia) for synchronized orders, stock, and tracking; automated shipping-label generation with tighter fulfillment links; analytics dashboards with KPI monitoring and alerting; and mobile access (PWA or native) to support on-the-go operations. Additional enhancements include real-time notifications with finer-grained RBAC, as well as broader evaluation through performance and security testing, usability studies with larger cohorts, and a subsequent corrective/adaptive maintenance phase during real-world deployment.

5. References

[1] We Are Social, "Global Digital Report 2025," 2025. [Online]. Available: https://wearesocial.com/

[2] DataReportal, "Digital 2025: Indonesia." Accessed: Oct. 07, 2025. [Online]. Available: https://datareportal.com/

[3] A. Waworuntu, "Rancang Bangun Aplikasi e-Commerce Dropship Berbasis Web," *Ultim. J. Tek. Inform.*, vol. 12, no. 2, pp. 118–124, 2020, doi: 10.31937/ti.v12i2.1823.

[4] A. De, "' Business on WhatsApp is tough now — but am I really a businesswoman ?' Exploring Challenges with Adapting to Changes in WhatsApp Business", doi: 10.1145/3706598.3713988.

[5] A. Aburub, A. Aburub, A. Aburub, A. G. Ben Hamida, and F. Khanfar, "The Impact of Dropshipping Service Quality on Customer Satisfaction in Saudi Arabia: The Mediating Role of Customer Trust. A Study on E-Commerce Users," *J. Posthumanism*, vol. 5, no. 4, pp. 1576–1598, 2025, doi: 10.63332/joph.v5i4.1292.

[6] N. Kim, B. Montreuil, and W. Klibi, “Inventory availability commitment under uncertainty in a dropshipping supply chain,” *Eur. J. Oper. Res.*, vol. 302, no. 3, pp. 1155–1174, 2022, doi: 10.1016/j.ejor.2022.02.007.

[7] B. Hakim and L. Maria, “Sistem Pengelolaan Transaksi In and Out Terintegrasi Berbasis Website,” *Progr. Stud. Sist. Informasi, Univ. Bunda Mulia*, vol. 17, no. 1, pp. 141–162, 2025.

[8] M. Saefudin, D. A. Megawaty, D. Alita, R. Arundaa, and E. Tenda, “Penerapan Framework Laravel Pada Sistem Informasi Posyandu Berbasis Website,” *J. Inform. dan Rekayasa Perangkat Lunak*, vol. 4, no. 2, pp. 213–220, 2023, doi: 10.33365/jatika.v4i2.2600.

[9] R. Darmawan and B. Hakim, “Perancangan Sistem Website E-Commerce Pada Pt. Natura Indoland Dengan Framework Codeigniter,” *JBASE - J. Bus. Audit Inf. Syst.*, vol. 5, no. 2, pp. 9–18, 2022, doi: 10.30813/jbase.v5i2.3776.

[10] B. Hakim *et al.*, “PERANCANGAN APLIKASI MANAJEMEN BENGKEL MOTOR BERBASIS IOS,” *Progr. Stud. Sist. Informasi, Univ. Bunda Mulia*, vol. 9, no. 3, pp. 5036–5044, 2025.

[11] A. B. Paksi, N. Hafidhoh, and S. K. Bimonugroho, “Perbandingan Model Pengembangan Perangkat Lunak Untuk Proyek Tugas Akhir Program Vokasi,” *J. Masy. Inform.*, vol. 14, no. 1, pp. 70–79, 2023, doi: 10.14710/jmasif.14.1.52752.

[12] P. Satria and G. Testiana, “Pemodelan Proses Pengoprasian Sistem Building Automation System (BAS) di Bandara SMB II Palembang Menggunakan Business Process Modelling Notation (BPMN),” pp. 151–158, 2025, doi: 10.31284/p.snestik.2025.7655.

[13] S. Ramdany, “Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web,” *J. Ind. Eng. Syst.*, vol. 5, no. 1, 2024, doi: 10.31599/2e9afp31.

[14] Rizki Ridwan, Nunu Kustian, and Erlin Windia Ambarsari, “Peran Data Store Dalam Mempresentasikan Hubungan Data Flow Diagram Ssadm Dengan Entity Relationship Diagram,” *J. Ilm. Tek. Mesin, Elektro dan Komput.*, vol. 2, no. 2, pp. 83–90, 2022, doi: 10.51903/juritek.v2i2.412.

[15] A. Praniffa *et al.*, “Pengujian Black Box Dan White Box Sistem Informasi Parkir Berbasis Web Black Box And White Box Testing Of Web-Based Parking Information System,” *J. Test. Dan Implementasi Sist. Inf.*, vol. 1, no. 1, pp. 1–16, 2023.

[16] T. Menora, C. H. Primasari, Y. P. Wibisono, T. A. P. Sidhi, D. B. Setyohadi, and M. Cininta, “Implementasi Pengujian Alpha dan Beta Testing Pada Aplikasi Gamelan Virtual Reality,” *KONSTELASI Konvergensi Teknol. dan Sist. Inf.*, vol. 3, no. 1, pp. 48–60, 2023, doi: 10.24002/konstelasi.v3i1.6625.

[17] U. Saputra, B. R. Nasution, A. A. Anggara, R. S. Qaisa, A. E. Jakfar, and N. Astrianda, “Analisa Pengujian Sistem Informasi Website E-Commerce Bali-Store Menggunakan Metode Black Box Testing,” *J. Teknol. Inf.*, vol. 2, no. 2, pp. 95–102, 2023, [Online]. Available: http://jurnal.utu.ac.id/JTI

[18] Y. Muharrama and A. Ginantaka, “Analisis Dan Desain Sistem Pengolahan Bioetanol Dari Sabut Kelapa Dengan Pendekatan Business Process Model And Notation (BPMN),” vol. 4, pp. 3256–3268, 2025.

[19] D. B. Yonathan and Y. M. Geasela, “Perancangan Sistem Transaksi Berbasis Website Menggunakan Metode Waterfall pada PT Josua,” *Adopsi Teknol. dan Sist. Inf.*, vol. 4, no. 1, pp. 37–63, 2025, doi: 10.30872/atasi.v4i1.2963.

[20] B. Hnatkowska and M. Cebinka, “ACTIVITY DIAGRAM GENERATION BASED ON USE-CASE TEXTUAL SPECIFICATION,” *Comput. Informatics*, vol. 38, pp. 1320–1340, 2019, doi: 10.31577/cai.

[21] M. Farid Al Jabbar, F. Adelina Harahap, and J. Immanuel Sijabat, “ANALISIS PERBANDINGAN PEMANFAATAN ERD UNTUK PROSES PEMBUATAN PROGRAM,” *J. Inform. Utama*, vol. 3, no. 1, pp. 12–22, 2025.