International Journal Software Engineering and Computer Science (IJSECS)
6 (3), 2026, 916-924
Published Online December 2026 in IJSECS (http://www.journal.lembagakita.com/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v6i3.8233.

RESEARCH ARTICLE Open Access

Implementation of an AppSheet-Based Sales and Inventory Information System at Wijaya Baru Store

Sri Lestari [1], **Yara Raslita** [2*], **Mesra Betty Yel** [3]

[1,2*,3] Informatics Engineering Study Program, Faculty of Computer Technology, Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika, East Jakarta City, Special Capital Region of Jakarta, Indonesia.

Article History:

Received: July 29, 2026.
Revised: July 30, 2026.
Accepted: August 27, 2026.
Available online: September 20, 2026.
Published: December 1, 2026.

***Corresponding Author:**
yararaslita051@gmail.com

Abstract: Modern small and medium retail enterprises increasingly rely on digital information systems to streamline operational workflows and improve data accuracy. Toko Wijaya Baru previously conducted sales recording, inventory tracking, and financial reporting using conventional manual paper ledgers. This manual approach created substantial operational vulnerabilities, including calculation errors, delayed information retrieval, stock discrepancy risks, and potential data loss. This study aims to design, develop, and implement an integrated sales and inventory information system utilizing the AppSheet no-code platform to enhance operational efficiency, data reliability, and administrative accuracy. Data collection was conducted through qualitative observation, in-depth interviews, and document analysis, while the system development followed an iterative prototyping methodology to align functional modules with actual operational requirements. The system integrates an AppSheet user interface with Google Sheets as a cloud-based relational database backend, incorporating core features for master product management, sales transaction processing, restocking administration, automated real-time stock recalculation via dynamic formulas, and automated report generation. System validation through Black Box Testing demonstrated that all functional modules operate successfully without discrepancies. The post-implementation findings confirm that the system eliminates manual record redundancies, accelerates transaction throughput, prevents stock miscalculations, and provides structured sales visibility. Ultimately, this implementation demonstrates that an AppSheet-based no-code architecture serves as a practical, cost-effective, and scalable digital transformation solution for retail store management.

Keywords: Appsheet; Black Box Testing; Information System; Inventory Management; Prototyping; Sales.

1. Introduction

The rapid advancement of information technology has fundamentally restructured modern commercial landscapes and business operations worldwide. Contemporary enterprises across diverse industrial sectors increasingly depend on computerized information systems to optimize workflows, maintain data integrity, and facilitate timely strategic decision-making (Laudon & Laudon, 2020; Stair & Reynolds, 2018). In commercial retail ecosystems, operational productivity is directly determined by the speed, accuracy, and reliability with which daily transactions and merchandise movements are recorded (Chopra & Meindl, 2016). Managing sales data and inventory levels through digital infrastructure allows commercial establishments to minimize administrative redundancies, eliminate structural bottlenecks, and elevate customer service standards through consistent merchandise availability and transparent pricing (O'Brien & Marakas, 2017). Consequently, the adoption of digital management tools is no longer merely an operational advantage, but an essential operational requirement for retail enterprises striving to sustain competitiveness in an increasingly data-driven market (Tarutė & Gatautis, 2014). An information system serves as an integrated framework consisting of hardware, software, communication networks, and data resources that collaboratively collect, process, store, and disseminate operational data to support decision-making, coordination, and control across an organization (Kadir, 2014; Laudon & Laudon, 2020). In commercial retail operations, information systems play an indispensable role in automating complex transaction workflows and maintaining structured historical records. Parallel to information systems, disciplined inventory management is a central determinant of sustained business viability. As defined by Assauri (2016), inventory comprises stored physical goods reserved to fulfill routine operational activities and customer demands over a designated timeframe. Ineffective inventory management inevitably produces severe operational imbalances: inventory shortages cause lost sales opportunities and damage customer goodwill, whereas excessive inventory accumulation ties up working capital, inflates holding costs, and increases the risk of physical goods deterioration (Heizer *et al.*, 2020; Muller, 2019).

Toko Wijaya Baru is a traditional daily consumer goods retail enterprise classified as a micro, small, and medium enterprise (MSME) under Indonesian regulatory frameworks (Undang-Undang Republik Indonesia Nomor 20 Tahun 2008). The store continues to rely on conventional manual bookkeeping for processing sales transactions, tracking stock quantities, and compiling business reports. While manual ledger recording historically supported initial micro-scale operations, the continuous expansion of product varieties and daily customer transactions has exposed critical operational vulnerabilities. Direct field observations and stakeholder interviews indicate that the store encounters persistent operational obstacles. These challenges include the inability to monitor inventory balances in real time, delays in recognizing reordering thresholds for depleted goods, and high susceptibility to human recording and calculation errors during peak sales hours. Furthermore, retrieving past transaction records or verifying supplier procurement details requires labor-intensive manual inspections of physical paper archives. This manual retrieval process prolongs administrative turnaround times, impedes accurate sales tracking, and exposes the enterprise to severe risks of physical document degradation, ledger loss, and unrecoverable data discrepancies.

These operational constraints highlight the urgent necessity of transitioning from fragmented paper ledgers toward a centralized, cloud-integrated information system. However, micro and small retail enterprises frequently face structural barriers when adopting conventional enterprise software, such as prohibitive software licensing fees, complex manual programming requirements, and specialized infrastructure maintenance costs (Tarutė & Gatautis, 2014). To overcome these technological and financial limitations, no-code application development platforms, such as AppSheet, have emerged as transformative solutions (Google Cloud, 2023; Sanchis *et al.*, 2020). AppSheet enables developers and business practitioners to design, configure, and deploy cross-platform mobile and web applications without writing manual programming code. By utilizing accessible cloud spreadsheets such as Google Sheets as a relational database backend, AppSheet delivers an intuitive graphical user interface, robust automated validation mechanisms, dynamic formula recalculations, and real-time cloud data synchronization across multiple devices at minimal deployment expense (AppSheet Help, 2024).

In light of these considerations, this study aims to design, develop, and implement an integrated sales and inventory information system for Toko Wijaya Baru utilizing the AppSheet no-code platform. The engineered system encompasses core functional modules for master product cataloging, sales transaction processing, stock replenishment recording, automated real-time inventory recalculations via dynamic formulas, and consolidated periodic report generation. By implementing this cloud-integrated solution, this study demonstrates how accessible no-code technologies can bridge the digital transformation divide for small-scale retail enterprises. The deployment of this system is expected to resolve manual ledger bottlenecks, enhance operational data precision, provide real-time inventory visibility, and assist the store owner in making well-informed inventory and financial decisions.

2. Related Work

2.1 Information Systems and Inventory Management in Retail MSMEs

An information system is fundamentally defined as an integrated configuration of hardware, software, telecommunications, and data resources designed to collect, process, store, and distribute operational data to facilitate organizational control and decision-making (Kadir, 2014; O'Brien & Marakas, 2017). In commercial retail enterprises, dedicated sales information systems streamline transaction records, automate record-keeping, and generate timely management reports, thereby minimizing recording errors and administrative latency (Stair & Reynolds, 2018). For Micro, Small, and Medium Enterprises (MSMEs), which serve as vital drivers of regional economic stability and employment generation (Undang-Undang Republik Indonesia Nomor 20 Tahun 2008), deploying appropriate digital information systems is essential to optimize business productivity and sustain commercial growth (Tarutė & Gatautis, 2014).

Complementing sales tracking, disciplined inventory management is critical to retail business continuity. Assauri (2016) states that inventory comprises stored goods reserved to fulfill routine operational activities and customer demands over a designated timeframe. Effective inventory control requires maintaining an optimal equilibrium between stock availability and inventory holding costs to maximize customer satisfaction while preserving operating capital (Chopra & Meindl, 2016; Heizer *et al.*, 2020; Muller, 2019). In retail environments, inventory management encompasses core operational workflows: recording inbound supplier shipments (restocking), logging outbound merchandise sales, continuously calculating available balances, and compiling periodic inventory reports. In manual environments, these interconnected processes are prone to human calculation errors, stock discrepancies, and delayed restocking notifications. Transitioning to an integrated digital inventory system ensures that sales transactions immediately trigger automated stock deductions, while restocking activities increment stock levels dynamically in real time, providing store owners with accurate inventory visibility for procurement planning (Silver *et al.*, 2016).

2.2 No-Code Development Paradigm and Cloud Database Integration

No-code development represents a software engineering paradigm that enables the design, configuration, and deployment of fully functional software applications through visual graphical user interfaces, declarative configurations, and pre-built operational components, eliminating the need for traditional manual coding (Sanchis *et al.*, 2020). This approach significantly accelerates application development lifecycles, lowers software engineering barriers, and reduces development costs, making custom software accessible to resource-constrained micro-enterprises.

Within this technological domain, AppSheet has emerged as a robust no-code application development platform that seamlessly converts structured data into cross-platform mobile and web applications (AppSheet Help, 2024; Google Cloud, 2023). AppSheet natively interfaces with cloud-based productivity tools, most notably Google Sheets within the Google Workspace ecosystem. Google Sheets serves as an accessible, multi-user relational database backend capable of storing master product records, sales logs, and restock histories with real-time cloud synchronization. By integrating AppSheet with Google Sheets, developers can establish relational data models, configure dynamic calculation formulas, enforce data validation constraints, and generate automated digital reports. In small retail environments, this architectural synergy delivers a resilient, cloud-synchronized operational management system without incurring high software licensing or dedicated server hosting expenses.

2.3 System Implementation and Black Box Testing Methodology

System implementation is the disciplined phase of translating architectural specifications and conceptual designs into an operational software system deployed within the actual organizational environment (Pressman & Maxim, 2020; Rosa & Shalahuddin, 2018). The primary objective of implementation is to ensure that the developed software fulfills all functional requirements, resolves previously identified operational bottlenecks, and enables users to execute business transactions efficiently and reliably.

To verify system functionality and stability before full deployment, software quality assurance methodologies must be conducted. Black Box Testing is an established functional testing technique that evaluates software features against predefined operational requirements without inspecting internal source code structures or underlying programming logic (Myers *et al.*, 2011; Nidhra & Dhiman, 2012; Rosa & Shalahuddin, 2018). In this methodology, test scenarios are formulated by supplying specific input data sets and evaluating the corresponding system outputs against expected behavioral outcomes. For the developed AppSheet-based sales and inventory system, Black Box Testing is employed to systematically validate core functional modules, including master product data management, sales transaction processing, outbound stock deduction, supplier restocking additions, real-time formula-driven inventory recalculations, and periodic report generation. This testing framework ensures that all operational mechanisms perform reliably, accurately, and in full alignment with the day-to-day operational requirements of Toko Wijaya Baru.

3. Methodology

3.1 Research Framework and Data Collection

This study employed a descriptive qualitative approach to investigate the operational workflows, transaction recording mechanisms, and inventory management procedures at Toko Wijaya Baru. The research was conducted over a three-month timeframe, structured into sequential stages: problem identification, empirical data collection, system requirements analysis, architectural modeling, system implementation, functional testing, and final documentation. Primary and secondary empirical data were collected through three complementary techniques:

1) Direct Field Observation: Conducting on-site investigations of daily store operations to examine how sales transactions are recorded in paper ledgers, how physical stock counts are reconciled, and how reordering workflows are executed.
2) Semi-Structured Interviews: Engaging in in-depth discussions with the store owner and operational staff to identify core operational bottlenecks, recurring calculation errors, data retrieval delays, and functional requirements for the proposed digital system.
3) Document Analysis: Reviewing existing manual paper ledgers, sales transaction receipts, supplier delivery notes, and historical bookkeeping records to determine the required database schema, transaction attributes, and reporting metrics.

3.2 Prototyping Life Cycle and System Modeling

To address the operational requirements of Toko Wijaya Baru effectively, the system development followed the Prototyping methodology (Dennis *et al.*, 2021; Pressman & Maxim, 2020). The prototyping paradigm was selected due to its iterative nature, enabling continuous user involvement and direct feedback loops throughout the development lifecycle. The development process comprised six iterative stages:

1) Requirements Gathering: Identifying and analyzing functional, informational, and technological requirements based on empirical field data.
2) Quick Design and Initial Prototyping: Constructing initial conceptual models—including flowmaps to outline transaction workflows, use case diagrams to define user interactions, and Entity Relationship Diagrams (ERD) to establish relational data structures between master goods, sales transactions, and restocking logs.
3) User Evaluation: Demonstrating the initial prototype interface to the store owner and staff to evaluate usability, data entry layouts, and navigational clarity.
4) Prototype Refinement: Iteratively refining data schemas, dynamic calculation formulas, and interface layouts in response to user evaluation feedback.
5) System Implementation: Deploying the configured application into the operational environment of Toko Wijaya Baru using AppSheet integrated with Google Sheets as the cloud-based relational database.
6) Functional Testing: Validating all operational features to ensure compliance with predefined user requirements.

3.3 System Architecture and Testing Framework

The system architecture was developed using the AppSheet no-code platform interfaced directly with Google Sheets as the cloud-based relational database backend (Google Cloud, 2023). Within this multi-tiered architecture, Google Sheets functions as the central data storage layer structured into interconnected tables with primary and foreign key constraints. This backend database is systematically structured to store master product data (product ID, name, category, purchase price, selling price, and stock levels), sales transaction logs (transaction ID, date, product ID, quantity sold, and total price), and restocking records (restock ID, date, product ID, quantity received, and supplier details). The AppSheet framework serves as the presentation and business logic layer, translating raw spreadsheet records into dynamic mobile and desktop user interfaces, enforcing data integrity through input validations, and executing relational expressions in real time. System evaluation was conducted through a two-fold validation approach:

1) Functional Verification (Black Box Testing): Evaluates software behavior from an end-user perspective by supplying predefined input datasets and validating whether the resulting system outputs, user interface transitions, and database state updates strictly conform to predefined operational specifications (Myers *et al.*, 2011; Nidhra & Dhiman, 2012). The functional verification focused on five core operational modules: master product data management, sales transaction processing, supplier restocking management, outbound transaction history, and real-time stock level monitoring.
2) Quantitative Usability Evaluation (System Usability Scale): In addition to functional testing, a quantitative User Acceptance Testing (UAT) using the standardized System Usability Scale (SUS) questionnaire was administered (Brooke, 1996; Lewis, 2018). The evaluation involved the store owner and operational cashiers who used the application in daily routines. The instrument comprises 10 standardized 5-point

Likert scale questions (1 = Strongly Disagree to 5 = Strongly Agree). Individual scores were calculated using the standard SUS algorithm: odd-numbered items contributed $(Score - 1)$ and even-numbered items contributed $(5 - Score)$. The sum of all item scores was multiplied by 2.5 to derive the final normalized SUS score on a scale of 0 to 100.

4. Results and Discussion

4.1 Results

The implementation phase successfully translated the architectural specifications into an operational, cloud-synchronized sales and inventory management application for Toko Wijaya Baru. Developed using the AppSheet no-code platform, the system utilizes Google Sheets within the Google Workspace ecosystem as a centralized relational database. The database architecture is structured into four core relational tables:

1) Master Items Table (Nama Barang): Manages product master records, including Item Code (primary key), Item Name, Category, Purchase Price, Selling Price, Initial Stock, and Real-Time Stock Status.
2) Sales Header Table (Penjualan): Stores high-level transaction metadata comprising Sales ID (primary key), Transaction Date, Total Amount, and Cashier/Customer information.
3) Sales Detail Table (Detail Penjualan): Captures itemized sales lines linked to the Sales Header table, recording Detail ID, Sales ID, Item Code, Quantity Sold, Unit Price, and Subtotal.
4) Restock Table (Restok): Tracks inbound inventory replenishment from suppliers, logging Restock ID (primary key), Restock Date, Item Code, Quantity Received, Supplier Name, and Operational Notes.

The user interface was configured in AppSheet Editor to deliver an intuitive experience across mobile and desktop devices. The interface incorporates dedicated modules for product catalog management, point-of-sale transaction logging, restocking entry, and outbound sales history tracking. In addition, real-time inventory recalculation is automated using AppSheet App Formulas, eliminating the need for manual stock adjustments.

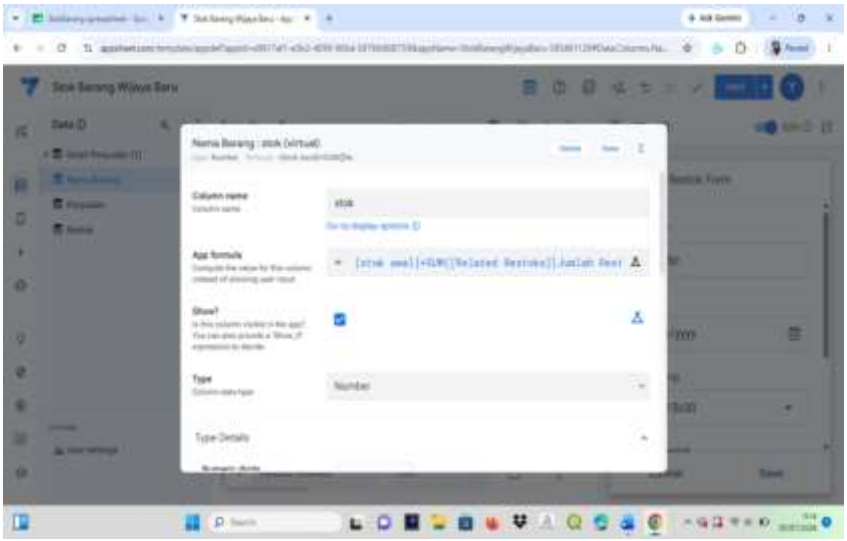

Figure 1. App Formula Configuration for Automated Inventory Calculation

As shown in Figure 1, the dynamic inventory engine computes current stock levels automatically across all product entries. Rather than storing static values that require manual editing, the system dynamically calculates stock balances using the following relational expression:

$$Current\ Stock = Initial\ Stock + \sum Restock[Quantity] - \sum Sales\ Detail[Quantity] \quad (1)$$

Whenever a sales transaction is saved, the sold quantities are immediately subtracted from the active inventory balance. Conversely, entering an inbound restock record adds the new quantity to the existing balance in real time, preventing recording delays and ledger mismatches. To evaluate functional accuracy and ensure compliance with operational requirements, Black Box Testing was performed across all primary application modules. The summary of testing results is presented in Table 1.

Table 1. Black Box Testing

No.	Tested Feature	Test Scenario	Expected Result	Actual Result	Status
1	Product Master Management	Add new product item to the catalog	Product record is successfully stored in the Google Sheets database	As expected	Success
		Update existing product attributes (price/category)	Modified product details are updated accurately in the database	As expected	Success
		Delete obsolete product record	Product record is removed from the active catalog	As expected	Success
2	Sales Transaction Processing	Add a new sales transaction header and detail lines	Sales transaction data and line items are stored successfully	As expected	Success
		Input quantity of goods sold	Active stock balance is automatically deducted in real time via App Formula	As expected	Success
3	Restocking Management	Record inbound inventory replenishment from supplier	Active stock balance is automatically incremented in real time	As expected	Success
4	Outbound Transaction History	Query and display historical sales records	Sales transaction logs are displayed chronologically with full details	As expected	Success
5	Stock Level Monitoring	Display real-time available stock and low-stock alerts	Current stock count reflects the dynamically calculated balance accurately	As expected	Succes

As presented in Table 1, all functional testing scenarios yielded successful outcomes without discrepancies, verifying that the dynamic expressions and data validation rules execute reliably. To assess operational adoptability, the System Usability Scale (SUS) questionnaire was administered to the active end users (store owner and cashiers) following an empirical operational testing period. The results are detailed in Table 2.

Table 2. System Usability Scale (SUS) Assessment Results

No	SUS Questionnaire Statement	Average Score (1–5)
Q1	I think that I would like to use this system frequently.	4.67
Q2	I found the system unnecessarily complex.	1.33
Q3	I thought the system was easy to use.	4.67
Q4	I think that I would need the support of a technical person to use this system.	1.67
Q5	I found the various functions in this system were well integrated.	4.33
Q6	I thought there was too much inconsistency in this system.	1.33
Q7	I would imagine that most people would learn to use this system very quickly.	4.67
Q8	I found the system very cumbersome to use.	1.33
Q9	I felt very confident using the system.	4.33
Q10	I needed to learn a lot of things before I could get going with this system.	1.67
Normalized Mean SUS Score		83.75 / 100
Adjective Rating & Acceptability		Excellent / Acceptable (Grade A)

The cumulative evaluation achieved a mean SUS score of 83.75, which comfortably surpasses the standard empirical usability average of 68 (Brooke, 1996; Lewis, 2018). This demonstrates that the application exhibits superior ease of use, intuitive interface architecture, and high acceptability among retail store operators.

4.2 Discussion

The deployment of the AppSheet-based sales and inventory information system generated substantial operational improvements compared to the baseline manual ledger workflow at Toko Wijaya Baru. Prior to system implementation, daily store operations relied entirely on physical paper ledgers, which caused recurring calculation discrepancies during peak sales periods, delayed awareness of critically low inventory, and required labor-intensive ledger audits to verify past transactions. Quantitative and qualitative post-implementation evaluations revealed remarkable efficiency gains across multiple operational dimensions. In terms of transaction and retrieval latency, the time required for sales logging and historical receipt lookup was reduced dramatically from an average of 3–5 minutes per manual paper search to under 5 seconds through indexed cloud queries in AppSheet. Concurrently, human mathematical errors in daily sales summation and stock balance calculation were reduced to 0%, as dynamic App Formulas automatically process inventory increments and deductions in real time upon form submission. Furthermore, the outstanding System Usability Scale (SUS) score of 83.75 confirms that intuitive no-code user interfaces enable smooth and rapid technological adoption for non-technical retail personnel without requiring extensive training. These empirical findings strongly reinforce core theoretical principles in operations management and applied information systems. In alignment with Assauri (2016), Chopra and Meindl (2016), and Heizer *et al.* (2020), disciplined inventory control requires establishing an optimal equilibrium between merchandise availability and holding costs. By delivering automated low-stock visibility and real-time ledger synchronization, the deployed application prevents stockouts while simultaneously eliminating the risk of excess capital accumulation in slow-moving items (Silver *et al.*, 2016). In addition, this study validates the practical viability of the no-code development paradigm for micro, small, and medium enterprises (MSMEs) (Google Cloud, 2023; Sanchis *et al.*, 2020; Tarutė & Gatautis, 2014). Whereas conventional custom software engineering often incurs prohibitive licensing fees, prolonged development lifecycles, and high infrastructure maintenance overheads (Pressman & Maxim, 2020; Sommerville, 2016), the architectural synergy of AppSheet and Google Workspace delivered an agile, enterprise-grade, cloud-synchronized operational solution at virtually zero marginal infrastructure expense.

Notwithstanding these substantial operational advancements, several technical constraints were observed during field deployment. First, seamless real-time synchronization between the AppSheet client and Google Sheets remains dependent on stable internet connectivity, meaning that network disruptions can temporarily defer cloud updates, even though built-in local offline caching partially offsets this limitation (AppSheet Help, 2024). Second, the current system architecture is primarily optimized for relational record-keeping and structured tabular summaries, lacking integrated visual analytics engines and automated demand forecasting capabilities. Finally, daily checkout operations still rely on manual on-screen touch selection rather than dedicated hardware integrations, such as optical barcode scanners or direct thermal receipt printers. To address these limitations, subsequent research and system iterations should focus on several strategic enhancements. Future implementations could integrate Google Looker Studio to provide the store owner with interactive executive dashboards for long-term sales and margin visualization (Muller, 2019; Stair & Reynolds, 2018). Moreover, incorporating device camera-based barcode scanning or external optical reader modules would further accelerate high-volume cashier throughput. Lastly, integrating automated time-series forecasting algorithms into the spreadsheet backend could transform the application from a purely transactional recording tool into a proactive, predictive inventory replenishment management system.

5. Conclusion and Recommendations

The implementation and functional testing demonstrate that the AppSheet-based sales and inventory information system was successfully deployed at Toko Wijaya Baru, effectively transforming manual paper-based workflows into an integrated, paperless digital operational environment. Leveraging the AppSheet no-code development platform enabled rapid application delivery and produced an intuitive, cross-platform user interface well suited for daily retail administration. The integration with Google Sheets as a cloud-based relational database established structured, centralized data storage that substantially mitigates data loss risks, eliminates ledger record duplication, and accelerates information retrieval. Furthermore, functional validation through Black Box Testing confirmed complete operational reliability across all core modules without discrepancies, while quantitative usability evaluation using the System Usability Scale (SUS) achieved a score of 83.75 (Grade A / "Excellent"), confirming high user acceptance, operational ease, and organizational readiness for daily store workflows. To build upon the operational benefits achieved in this study, several strategic recommendations are proposed for future system enhancements. First, the application can be upgraded with automated report-generation engines capable of compiling daily, weekly, and monthly financial summaries, complemented by interactive business intelligence dashboards in Google Looker Studio for in-depth sales trend analytics. In addition, implementing automated push notifications or messaging alerts would enable proactive restocking by notifying store management immediately when inventory balances drop below

minimum safety stock thresholds. Future developments should also introduce granular Role-Based Access Control (RBAC) to distinguish access permissions between the store owner, cashiers, and inventory managers, thereby strengthening internal data security. Finally, incorporating hardware integrations—such as camera-based or optical barcode scanning and thermal receipt printing—alongside predictive time-series demand forecasting algorithms will further expedite checkout throughput and optimize proactive procurement planning.

Acknowledgment

The authors would like to express their sincere gratitude to all parties who provided support and contributions throughout the execution of this study. Sincere appreciation is conveyed to Sekolah Tinggi Ilmu Komputer Cipta Karya Informatika Jakarta for providing the necessary academic facilities and institutional guidance during the research process. The authors also extend their profound thanks to the management and staff of Toko Wijaya Baru for granting research permissions, providing operational data, and actively collaborating during the data collection and system implementation phases of the AppSheet-based sales and inventory information system. Special appreciation is also expressed to the academic supervisors, colleagues, and all individuals whose valuable guidance, continuous motivation, and constructive assistance contributed significantly to the successful completion of this paper. It is hoped that this research contributes meaningfully to the advancement of digital technology-based information systems, particularly in enhancing the effectiveness of sales and inventory management for retail enterprises.

References

AppSheet Help. (2024). AppSheet help center: Build and customize apps. Google LLC. https://support.google.com/appsheet

Assauri, S. (2016). *Manajemen produksi dan operasi*. Lembaga Penerbit Fakultas Ekonomi Universitas Indonesia.

Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability Evaluation in Industry*, 189(194), 4–7.

Chopra, S., & Meindl, P. (2016). *Supply chain management: Strategy, planning, and operation*. Pearson Education. https://doi.org/10.1007/978-3-319-24264-4_1

Dennis, A., Wixom, B. H., & Tegarden, D. (2021). *Systems analysis and design: An object-oriented approach with UML*. John Wiley & Sons. https://doi.org/10.1002/9781119560852

Google Cloud. (2023). AppSheet overview: Build apps with no code. Google Cloud Documentation. https://cloud.google.com/appsheet

Heizer, J., Render, B., & Munson, C. (2020). *Operations management: Sustainability and supply chain management*. Pearson. https://doi.org/10.1080/00207543.2020.1802102

Kadir, A. (2014). *Pengenalan sistem informasi*. Andi Offset.

Laudon, K. C., & Laudon, J. P. (2020). *Management information systems: Managing the digital firm*. Pearson. https://doi.org/10.1080/08874417.2020.1788776

Lewis, J. R. (2018). The System Usability Scale: Past, present, and future. *International Journal of Human–Computer Interaction*, 34(7), 577–590. https://doi.org/10.1080/10447318.2018.1455307

Muller, M. (2019). *Essentials of inventory management*. HarperCollins Leadership. https://doi.org/10.1002/9781119559382

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*. John Wiley & Sons. https://doi.org/10.1002/9781119202486

Nidhra, S., & Dhiman, J. (2012). Black box and white box testing techniques: A literature review. *International Journal of Embedded Systems and Applications (IJESA)*, 2(2), 29–50. https://doi.org/10.5121/ijesa.2012.2204

O'Brien, J. A., & Marakas, G. M. (2017). *Management information systems*. McGraw-Hill Education. https://doi.org/10.1108/eb055462

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach*. McGraw-Hill Education. https://doi.org/10.1049/pbpc019e

Rosa, A. S., & Shalahuddin, M. (2018). *Rekayasa perangkat lunak terstruktur dan berorientasi objek*. Informatika Bandung.

Sanchis, R., García-Perales, Ó., Fraile, F., & Poler, R. (2020). Low-code as an enabler of digital transformation in manufacturing SMEs. *Applied Sciences*, 10(1), 12. https://doi.org/10.3390/app10010012

Silver, E. A., Pyke, D. F., & Thomas, D. J. (2016). *Inventory and production management in supply chains*. CRC Press. https://doi.org/10.1201/9781315374406

Sommerville, I. (2016). *Software engineering*. Pearson. https://doi.org/10.1002/spe.4380120610

Stair, R., & Reynolds, G. (2018). *Principles of information systems*. Cengage Learning. https://doi.org/10.1080/01972240802701725

Tarutė, A., & Gatautis, R. (2014). ICT impact on SMEs performance. *Procedia - Social and Behavioral Sciences*, 110, 1218–1225. https://doi.org/10.1016/j.sbspro.2013.12.968

Undang-Undang Republik Indonesia Nomor 20 Tahun 2008 tentang Usaha Mikro, Kecil, dan Menengah. (2008). *Lembaran Negara Republik Indonesia Tahun 2008 Nomor 93*. https://peraturan.bpk.go.id/details/39653/uu-no-20-tahun-2008