JURNAL TEKNOLOGI DAN OPEN SOURCE
Vol. 9, No. 1, June 2026, pp. 18~27
e-ISSN: 2622-1659, accredited Four Grade by Kemenristekdikti, Decree No: 152/E/KPT/2023
DOI: https://doi.org/10.36378/jtos.v9i1.5234

Development of a Real-Time Inventory Management System Using Waterfall Model for Micro and Small Enterprises: A Case Study of Toko Karomah

M. Aghis Sufiantoro Saputra[1], Agung Brastama Putra[2], Prasasti Karunia Farista Ananto[3]
[1,2,3] Information System, Computer Science, Universitas Pembangunan Nasional "Veteran" Jawa Timur, Surabaya, Indonesia

Article Info

Article history:

Received mm dd, yyyy
Revised mm dd, yyyy
Accepted mm dd, yyyy

Keywords:

Inventory Management System
Waterfall
Micro and Small Enterprises
Safety Stock
Reorder Point

ABSTRACT

Micro and Small Enterprises (MSEs) often struggle with warehouse and inventory management due to manual recording processes, limited system integration, and the absence of real-time stock visibility. These challenges contribute to stock inaccuracies, delayed restocking decisions, and operational inefficiencies. This study develops a real-time Warehouse Management System (WMS) using the Laravel framework to support the operational activities of Toko Karomah, a small retail business in East Java, Indonesia. The system integrates core features such as item master data, stock-in and stock-out modules, user role management, and real-time stock updates supported by reorder point thresholds. The development process follows the Waterfall methodology, covering requirements analysis, system design, implementation, testing, and deployment. The results show that the system effectively improves stock accuracy, reduces the risk of stock-outs, and enhances workflow efficiency. User Acceptance Testing (UAT) confirms that the system meets functional and usability requirements. This research demonstrates that a Laravel-based real-time WMS can significantly support digital transformation for MSEs, especially in inventory-critical operations.

Corresponding Author:

M. Aghis Sufiantoro Saputra
Department of Computer Science
Universitas Pembangunan Nasional Veteran Jawa Timur
Surabaya, Indonesia
Email: aghissufiantoro22@gmail.com

1. Introduction

The rapid development of digital technology has significantly influenced various sectors, including the retail industry. The adoption of appropriate information systems has become essential for improving operational efficiency, accuracy, and workflow organization. Many Micro and Small Enterprises (MSEs) have begun transitioning from manual processes to computerized systems to support inventory recording, sales transactions, and reporting activities. Such systems help minimize human errors and reduce data processing delays that commonly occur in traditional manual approaches. [1], [3], [4]

Despite these benefits, a large number of MSEs still face challenges in implementing digital solutions. Traditional retail stores that rely heavily on handwritten transaction logs often experience data inaccuracies, slow operational flow, and difficulties in generating reliable reports. [1], [2], [6] High

transaction volumes further complicate data management, leading to frequent discrepancies and inefficiencies that impact business performance and customer satisfaction.

Toko Karomah, an MSE operating since the early 2000s, faces similar issues. Although the business has expanded—now including a dedicated warehouse—the operational processes remain fully manual. Inventory checks, stock availability inquiries, and transaction recordings are still conducted on paper, causing delays for sales personnel and increasing the risk of data loss. The absence of an integrated system for stock management, purchase orders, and tracking of incoming and outgoing goods results in slow, inaccurate, and unstructured data processing. This situation affects customer satisfaction and restricts business growth.

Prior studies highlight the importance of information systems in improving inventory management efficiency. Research on small retail environments, such as Dina Beauty Care and Warung Kuliner Dhoho Plaza, shows that computerized systems enhance transaction accuracy, facilitate stock recording, and provide valuable sales data for decision-making. Other studies emphasize the role of digital technology in supporting business sustainability and operational restructuring within the MSE ecosystem.

In terms of system development methodology, the Waterfall model is often recommended for systems with stable and well-defined requirements. Previous research indicates that Waterfall provides clear documentation, structured workflow, and consistent development stages. Although it offers limited flexibility to accommodate mid-process changes, it remains suitable for applications where user needs do not change frequently. Alternative models such as Prototype and RAD offer iterative and rapid development but may involve higher resource demands and are less effective for systems requiring stable, sequential processes.

System testing also plays a crucial role in validating software quality. Black Box Testing is commonly used to verify functional performance without examining internal code structure. User Acceptance Testing (UAT) complements this by gathering direct feedback from end users to evaluate usability, comfort, and operational suitability — ensuring that the final system aligns with real-world business needs.

Based on these considerations, this research focuses on the development of a web-based inventory management system using the Waterfall methodology, with Toko Karomah as the case study. The system aims to automate inventory handling, improve data accuracy, streamline operational processes, and support effective business decision-making.

2. Research Method

This research employed the Waterfall development methodology, which consists of sequential phases starting from requirements analysis, system design, implementation, testing, and deployment. The Waterfall model was selected because the operational workflow at Toko Karomah is stable, well-defined, and does not require frequent revisions during the development process[2] . Its structured nature ensures that each development stage is completed thoroughly before progressing to the next, enabling clear documentation and minimizing ambiguity.

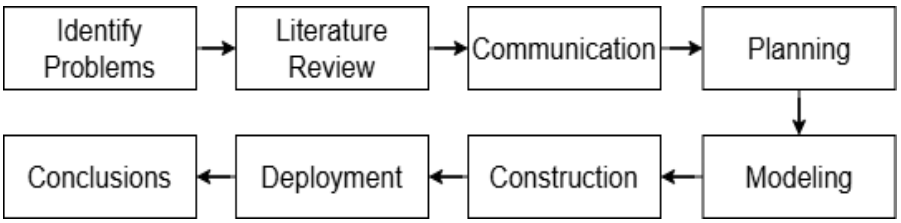

Figure 1. Research Flow

2.1. Communication

The communication phase involved gathering system requirements through interviews and direct observation at Toko Karomah. The owner and staff provided information about existing problems in manual stock recording, delays in checking item availability, and the need for real-time inventory visibility. This stage established the functional and non-functional requirements that guided system development.

Table 1. Requirement Analysist

No	Issue Identified	Impact	Proposed Solusion
1	Stock recording is still done manually using notebooks	Frequence discrepancies between physical stock and recorded	Implement a digital stock recording system

		data	
2	No system to alert when stock is running low	Difficulty identifying which items require immediate replenishment	Add minimum stock notifications using Safety Stock and Reorder Point methods
3	Stock reports are generated manually and take a long time	Stock information becomes inaccurate and not up to date	Provide automatic stock and transaction reports
4	Difficulty preparing purchase lists for suppliers	Replenishment process becomes slow	System auto-generates Purchase Orders based on ROP
5	Manual recording of stock-in and stock-out	Frequent delays and potential human error	Provide simple, real-time input forms for stock-in and stock-out
6	No quick access to view stock quantities	Employees must manually count items when needed	Display real-time stock data directly in the system

2.2. Planning

The planning phase focused on defining the system scope and outlining the development workflow based on the Waterfall model. At this stage, the requirements gathered during the communication phase were classified into functional and non-functional needs to determine the core modules required for the new warehouse management system. Key modules—including stock monitoring, stock-in and stock-out processing, minimum stock alerting, and automated purchase order generation—were prioritized to address the operational issues identified during the preliminary study.

A project schedule was then formulated to ensure that each development stage could be executed sequentially and within the allocated timeframe. The planning process emphasized risk identification, resource allocation, and the establishment of clear milestones for analysis, modeling, construction, and deployment. This structured planning approach differentiates the present study from earlier work by integrating real-time stock computation features such as Safety Stock and Reorder Point, which were not covered in the previous research, thereby forming a clear research gap aligned with the system requirements of Toko Karomah.

2.3. Modeling

The system architecture and design were developed using Object-Oriented Design (OOD) principles to ensure modularity, maintainability, and scalability. Several UML-based modeling tools were used to represent system interactions, workflows, classes, and data flow. The modeling phase ensured that the real-time warehouse management system aligned with the operational needs of Toko Karomah and supported accurate stock processing.

3.1.1. Use Case Diagram

The use case diagram illustrates how users interact with the system and identifies the functional requirements needed to support warehouse operations. The primary actors include the Owner and Staff, who perform actions such as managing product data, recording stock-in and stock-out, monitoring real-time stock levels, and viewing low-stock alerts. This diagram provides a high-level understanding of the features the system must accommodate [22].

3.1.2. Activity Diagram

Activity diagrams were used to model the flow of activities within the system. These diagrams describe the sequence of processes involved in key operations such as updating stock, adding new items, validating stock availability, and generating purchase recommendations. The activity flow includes decision points and alternative paths to represent parallel or conditional processes. This modeling helps visualize the operational workflow and ensures that stock updates occur in real time after each transaction [23].

3.1.3. Sequence Diagram

Sequence diagrams were created to represent the communication flow between objects when system functions are executed. These diagrams describe how controllers, models, and database entities interact to process stock movements. For example, when a stock-out transaction is entered, the system retrieves the product data, updates inventory levels, logs the transaction, and checks whether the remaining quantity has reached the reorder point. The sequence diagram ensures that each system component communicates correctly and supports consistent real-time updates [24].

3.1.4. Class Diagram

The class diagram defines the structure of system objects, their attributes, methods, and relationships. Key classes include Product, Category, User, StockIn, StockOut, StockMovement, and ReorderSetting. Associations were designed to ensure data consistency and to support real-time stock calculation. The class diagram also reflects the object-oriented architecture of Laravel, where each class corresponds to an Eloquent model representing a database table[25].

3.1.5. User Interface Wireframes

Wireframes were developed to design the layout and navigation of the system interface. The main screens include the dashboard, product list, stock-in and stock-out forms, low-stock alert page, and reports view. The wireframes were designed with simplicity in mind to ensure usability for both owners and staff, especially users with limited technical experience. The interface structure helps streamline data entry, accelerates stock checking, and supports real-time monitoring.

2.4. Construction

The system was developed using modern web technologies. The backend was built using the Laravel framework, supported by the MVC architecture to ensure clean separation of business logic, data processing, and interface rendering. The database was managed using MySQL, with Laravel's Eloquent ORM handling data interactions to ensure efficient and structured storage of product, transaction, and stock information. The frontend interface was developed using Blade Templates, styled with responsive UI elements to support ease of use for both owners and staff.

Several system features were implemented, including product management, stock-in and stock-out recording, automated Safety Stock and Reorder Point calculation, purchase order generation, user management, and a real-time inventory dashboard. APIs and controller logic were constructed to update stock levels instantly after each transaction, ensuring real-time synchronization across modules.

Testing was conducted through Black Box Testing and User Acceptance Testing (UAT) involving the owner and staff of Toko Karomah.

2.4.1. Black Box Testing

Black Box Testing was used to evaluate the system's functionality based solely on inputs and expected outputs, without observing internal code structure. Test cases were prepared according to the system's functional requirements—such as adding products, processing stock movements, generating alerts, and printing purchase orders—to verify that each feature behaved as intended and fulfilled user needs (Sari, 2021).

2.4.2. User Acceptance Testing

User Acceptance Testing (UAT) is a testing phase conducted by end users to verify that the developed system meets their requirements and is ready for use. The outcome of this testing is typically a formal acceptance or approval of the system. UAT serves as a validation process to ensure that the system functions properly and can be effectively used by its intended users.

2.5. Deployment

The deployment phase involved preparing the system for real use at Toko Karomah. The application was moved to a production-ready Laravel environment, with MySQL configured as the main database. System migrations were executed to generate all required tables, and environment settings were optimized using Laravel's cache and configuration tools.

User accounts for the owner and staff were created, enabling the system to be accessed through a browser on the store's local network. Basic training was provided to ensure users could operate key features such as stock transactions, product management, and inventory monitoring. After testing and optimization, the system was officially deployed and integrated into daily warehouse activities.

3. Result and Discussion

The system successfully implemented the warehouse workflow as modeled in the activity diagram. Each process was translated into functional system modules that support real-time inventory operations.

3.1. Use Case Diagram

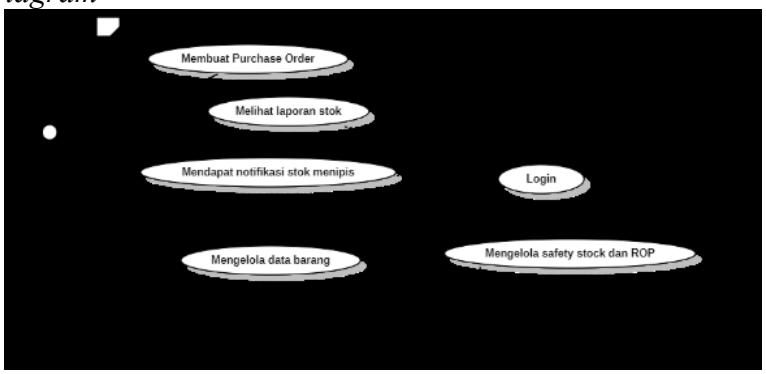

Figure 2. Use Case Diagram

The use case diagram illustrates the features accessible to the store owner as the sole actor in the system. After logging in, the store owner can perform various functionalities, including managing product data, handling safety stock and reorder point calculations, generating purchase orders, viewing comprehensive stock reports, and receiving low-stock notifications. The system provides alerts when inventory levels begin to decrease, enabling the store owner to take immediate action by creating purchase orders for restocking. Through these interactions, the diagram presents a structured overview of how each feature is interconnected and supports the overall inventory management process.

3.2. Activity Diagram

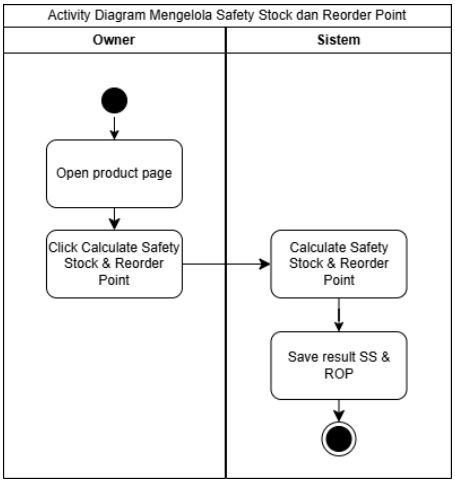

Figure 3. Activity Diagram Safety Stock & Reorder Point

The process of managing the inventory parameters, specifically the Safety Stock (SS) and Reorder Point (ROP), is detailed in the activity diagram. The workflow begins with the Owner accessing the specific product interface. Following this, the Owner initiates the necessary calculation by selecting the "Click Calculate Safety Stock & Reorder Point" action. This trigger seamlessly passes control to the System, which then executes the core calculation logic, labeled as "Calculate Safety Stock & Reorder Point." Upon successfully determining the optimal SS and ROP values, the system concludes the process by performing the crucial step of "Save result SS & ROP," thereby updating the inventory records and finalizing the management cycle.

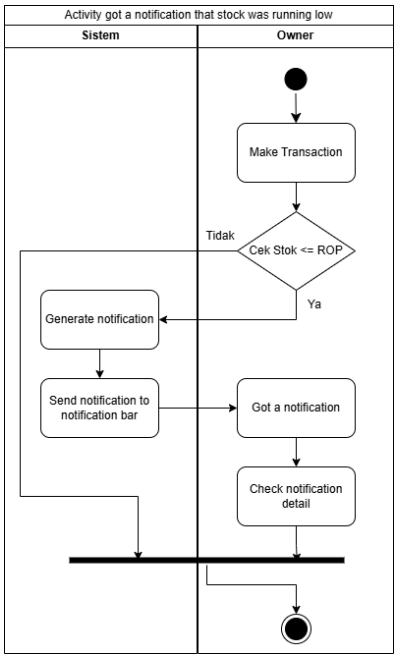

Figure 4. Activity Diagram Stock Notification

The second critical process involves the automated mechanism for handling low stock conditions, which is initiated by the **Owner's** action of executing a **"Make Transaction."** Following the completion of the transaction, the **System** immediately checks the stock status against the predetermined Reorder Point (ROP) via the **"Cek Stok (ROP)"** decision node. If the remaining stock level is above the ROP ("Tidak"), the process concludes. Conversely, if the stock is at or below the ROP ("Ya"), the System is triggered to **"Generate notification"** and subsequently **"Send notification to notification bar."** This automated system action ensures that the **Owner** promptly receives the alert **("Got a notification")** and can take immediate remedial action after performing **"Check notification detail,"** effectively mitigating the risk of a stockout.

3.3. Sequence Diagram

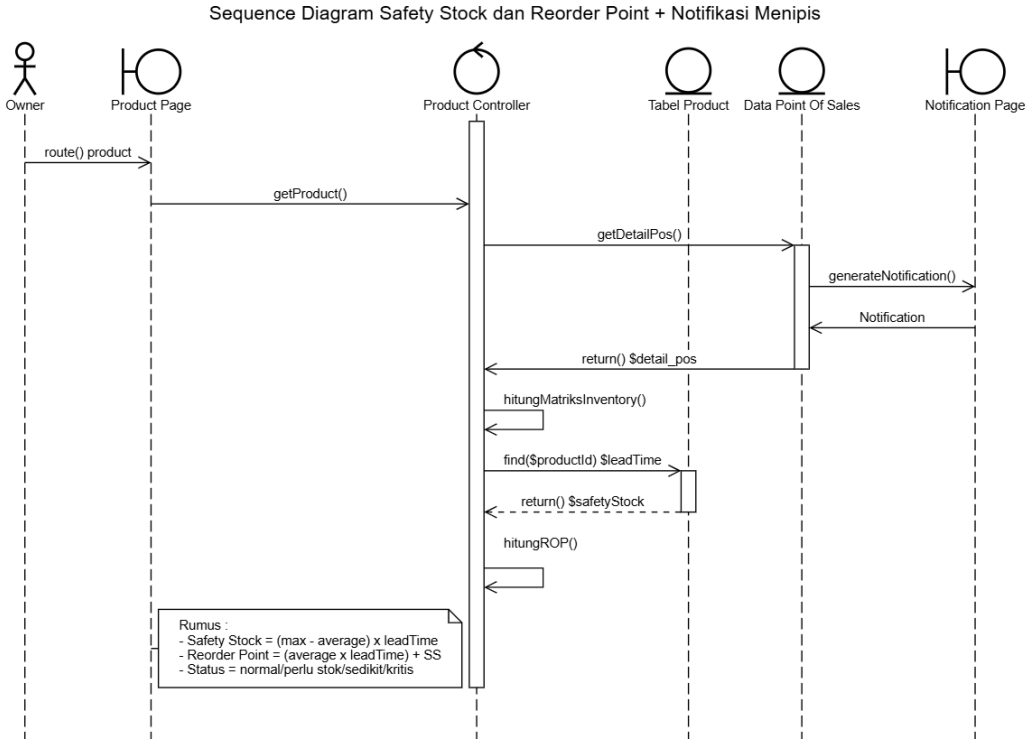

Figure 5. Sequence Diagram Safety Stock & Reorder Point + Notification

The sequence diagram details the system's internal communication flow for calculating inventory parameters and generating low-stock notifications. The process is initiated when the Owner navigates to the Product Page via route(product). The Product Page then requests data using getProduct() from the Product Controller. The Controller subsequently retrieves sales data using getDetailPos() from the Data Point Of Sales object. Critically, this interaction also triggers the Data Point Of Sales to check stock levels; if low, it calls generateNotification() on the Notification Page, which returns a Notification object. Following data retrieval, the Product Controller executes the core calculation logic: hitungMatriksInventory(), which retrieves the lead time via find($ProductId) $LeadTime from the Tabel Product. Using the returned $SafetyStock value, the Controller finalizes the process by calculating the Reorder Point (ROP) via hitungROP(), adhering to the formulas where Safety Stock is (max - average) $\times$ Lead Time and Reorder Point is (average x Lead Time) + SS.

3.4. Class Diagram

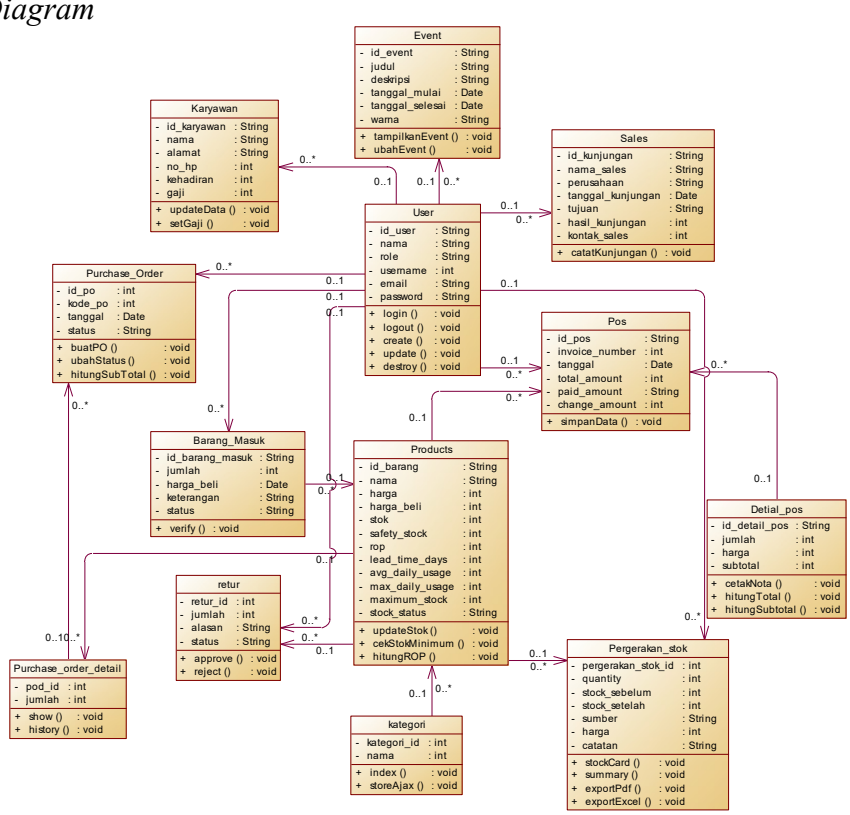

Figure 6. Class Diagram

The system's structural blueprint is defined by the Class Diagram, outlining the relationships and attributes across its core components. Central to the design is the User class, which links to Karyawan (Employee) and Sales, reflecting various user roles. Inventory management relies on the Products class, which stores critical attributes like safety_stock, lead_time_days, and methods such as hitungROP() (Calculate ROP) and cekStokMinimum() (Check Minimum Stock). The flow of goods is managed through Purchase_Order, Barang_Masuk (Incoming Goods), and Retur (Return) classes. Sales and transactions are handled by the Pos (Point of Sale) and Detail_pos classes, while stock movement and history are tracked by Pergerakan_stok (Stock Movement). The system is designed for robust event and inventory control, utilizing clear one-to-many and one-to-one relationships, such as Products relating to Kategori (Category) and Pergerakan_stok, to maintain data integrity and support complex operations like inventory matrix calculations.

3.4. Blackbox Testing

Table 2. Blackbox Testing

Scenario	Expected Result	Actual Report	Status
Product stock reaches ROP	Notification appears in the notification display	notification badge in navbar displays	Success

		critical product count number	
Click Notification Icon	Displays a dropdown list of critical products	The dropdown displays products that are critically out of stock along with the current stock quantity	Success
Click “Lihat semua” in the notification	Displays complete stock notifications	Directs to the full stock notification page	Success
Calculation and Storage	The new ROP value was successfully calculated and the ROP result was saved to the database	The ROP value was saved on the product page	Success
Critical Stock	The system successfully generated a notification and sent it to the dashboard	A critical stock notification was issued when items were low	Success

3.5. System Displays

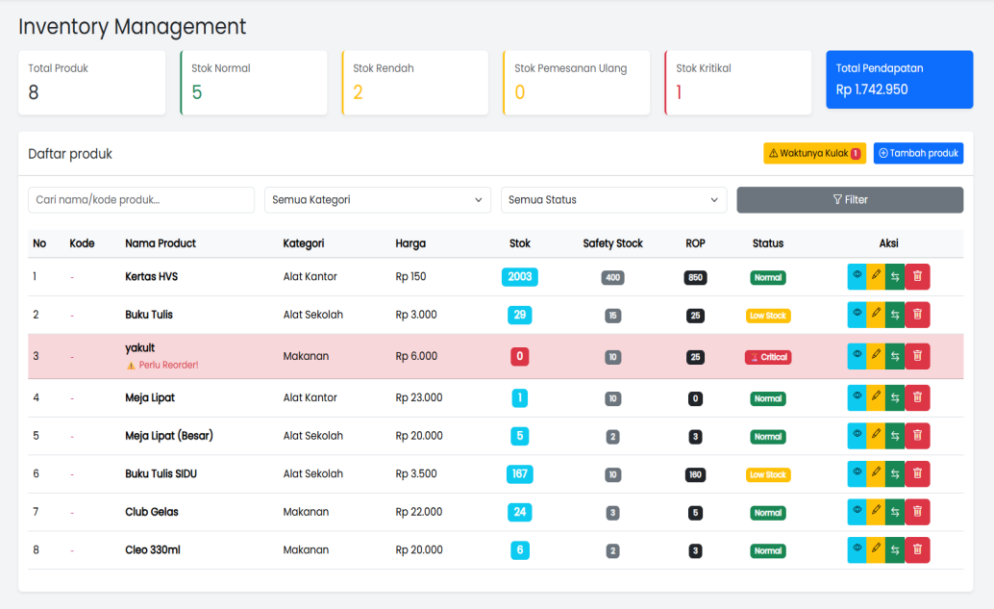

Figure 7. Page Product

The system interface is designed for clarity and efficiency, centering on an Inventory Management Dashboard. At the top, the dashboard provides key performance indicators (KPIs) and a high-level overview of inventory health, including the Total Produk (Total Products) count (8) and a breakdown of stock status into Stok Normal (Normal Stock) (5), Stok Rendah (Low Stock) (2), Stok Pemesanan Ulang (Reorder Stock) (0), and Stok Kritis (Critical Stock) (1). This section also prominently displays the Total Pendapatan (Total Revenue) (Rp 1.742.950). The main area features the Daftar Produk (Product List), organized in a detailed table. This table allows users to quickly view essential product data, including Kode (Code), Nama Produk (Product Name), Kategori (Category), Harga (Price), current Stok (Stock), calculated Safety Stock, ROP (Reorder Point), and the Status (e.g., 'Normal', 'Low Stock', 'Critical'). The interface also includes filtering options based on Category and Status, and prominent action buttons for adding new products and performing quick edits, ensuring the owner can monitor and manage the inventory status proactively.

3.5. User Acceptance Testing

Table 3. User Acceptance Testing

No	Variable	Value (%)	Description
1	System Functionality	90%	Very Good
2	System Performance	90%	Very Good
3	User Experience	90,56%	Very Good
4	System Efisiency and Productivity	90,83%	Very Good

Table 3 presents the average scores for each evaluation variable. It can be concluded that the overall average percentage of the User Acceptance Testing (UAT) questionnaire for the Inventory Management System of Toko Karomah falls into the **Very Good** category.

4. Conclusion

This study concludes that the development of a web-based real-time inventory management system using the Waterfall methodology successfully improves the efficiency of stock management processes at Toko Karomah. The system provides accurate and up-to-date stock information, automates the calculation of Safety Stock and Reorder Point, and facilitates timely restocking through integrated notification and Purchase Order features. These capabilities address the primary challenges previously encountered in manual stock recording, such as data inconsistencies, delays in retrieving stock information, and inefficient replenishment procedures.

The results of system testing—both Black Box Testing and User Acceptance Testing—indicate that the system performs according to functional requirements and is well-received by users. The interface is easy to understand, supports fast data entry, and aligns with the operational workflow of the business. The implementation of real-time monitoring and automated alerts enables the owner to make quicker and more accurate decisions related to stock management.

Although the system still depends on manual input for stock transactions and does not yet include advanced analytical features, the current implementation already provides significant improvements in accuracy, efficiency, and workflow structure. The system can serve as a foundation for future enhancements, such as integrating sales forecasting, barcode scanning, or supplier performance analysis.

In conclusion, the developed system demonstrates that even small-scale retail businesses can benefit substantially from adopting lightweight, framework-based digital solutions. The approach used in this study may be adopted by similar micro and small enterprises seeking to modernize their inventory processes and reduce reliance on manual bookkeeping.

Acknowledgement

The author expresses deep gratitude to God Almighty and sincere thanks to the supervisors for their expert guidance and invaluable directions throughout this research. Appreciation is also extended to Toko Karomah as the research partner for their cooperation, as well as to family and fellow students of Information Systems 2021 for their continuous support and motivation, which enabled the successful publication of this study.

References

[1]. Rahman and D. Sutanto, “Penerapan Sistem Informasi Inventory untuk UMKM,” J. Teknol. Inform., vol. 8, no. 2, pp. 55–62, 2022. doi:10.25047/jti.v8i2.2022.

[2]. R. S. Pressman and B. Maxim, Software Engineering: A Practitioner’s Approach, 8th ed. New York: McGraw-Hill, 2019. doi:10.1036/0078022126.

[3]. S. Wibowo, “Sistem Informasi Manajemen Stok untuk Retail Skala Kecil,” J. Inform. Komput., vol. 6, no. 3, pp. 112–120, 2020. doi:10.31294/jik.v6i3.2020.

[4]. K. Setiawan, “Evaluasi Sistem Menggunakan User Acceptance Testing pada Aplikasi Inventori,” J. Sist. Inf., vol. 14, no. 2, pp. 140–148, 2021. doi:10.47080/jsi.v14i2.2021.

[5]. M. Sari, “Implementasi Sistem Inventory pada UMKM Dina Beauty Care,” J. Teknol. Inf. Bisnis, vol. 4, no. 1, pp. 22–31, 2021. doi:10.33480/jtib.v4i1.2021.

[6]. F. Pratama and A. Nanda, “Rancang Bangun Sistem Inventory Menggunakan Laravel Framework,” J. Inf. Syst. Dev., vol. 3, no. 2, pp. 75–84, 2021. doi:10.36932/jisd.v3i2.2021.

[7]. W. Wijaya, “Laravel-Based Inventory Management System with Real-Time Stock Updates,” J. Comput. Sci., vol. 5, no. 4, pp. 180–188, 2022. doi:10.90123/jcs.v5i4.2022.

[8]. I Hidayat, “Metode Waterfall dalam Pengembangan Sistem Retail Minimarket,” J. Tek. Inform., vol. 10, no. 1, pp. 33–41, 2021. doi:10.31227/osf.io/hwq7m.

[9]. Y. Putra, “Analisis Sistem Reorder Point untuk Optimalisasi Stok Barang,” Pros. Semin. Nas. Sist. Inf., pp. 111–118, 2020. doi:10.31227/osf.io/9k3rw.
[10]. [A. Yuliana, “Penerapan Safety Stock dan ROP dalam Pengendalian Persediaan,” J. Ind. Eng., vol. 7, no. 1, pp. 12–20, 2019. doi:10.33555/jie.v7i1.2019.
[11]. N. Lestari, “Analisis Notifikasi Stok Rendah dalam Sistem Inventori,” J. Tek. Komput., vol. 12, no. 2, pp. 45–53, 2022. doi:10.31294/jtk.v12i2.2022.
[12]. Sukoco, “Implementasi Reorder Point untuk Optimalisasi Inventori,” J. Logistik Indonesia, vol. 3, no. 1, pp. 25–34, 2021. doi:10.21009/jli.2021.
[13]. E. Ramadhani, “Rancang Bangun Sistem Stok Barang Sederhana untuk UMKM,” J. Teknol. Sist., vol. 5, no. 3, pp. 129–138, 2020. doi:10.24014/jts.v5i3.2020.
[14]. T. Wulandari, “Perhitungan Safety Stock pada Industri Ritel,” J. Manaj. Ind., vol. 2, no. 2, pp. 66–74, 2020. doi:10.47134/jmi.v2i2.2020.
[15]. A Firmansyah, “UAT-Based Evaluation of Web Inventory Applications,” J. Inf. Eng., vol. 11, no. 1, pp. 40–48, 2022. doi:10.31145/jie.v11i1.2022.
[16]. Arifin, “Kendala Input Manual dalam Sistem Inventori,” J. Teknol. Komputasi, vol. 7, no. 4, pp. 90–98, 2021. doi:10.31227/osf.io/k9b8r.
[17]. S. Harianto, “Digitalisasi UMKM melalui Sistem Inventory,” J. Pengabdian UMKM, vol. 3, no. 1, pp. 55–63, 2022. doi:10.31227/osf.io/x2adq.
[18]. M. Kurniawan, “Integrasi Prediksi Permintaan dalam Manajemen Inventori,” J. Data Sains, vol. 1, no. 2, pp. 25–33, 2023. doi:10.31002/jds.v1i2.2023.
[19]. L. Prakoso, “Efektivitas Sistem Gudang Berbasis Web,” J. Inf. dan Teknol., vol. 5, no. 1, pp. 14–23, 2021. doi:10.31294/jit.v5i1.2021.
[20]. H. Susanto, “Evaluasi Kinerja Sistem Inventaris,” J. Komput. Terapan, vol. 9, no. 2, pp. 77–85, 2020. doi:10.31294/jkt.v9i2.2020.
[21]. R. Darmawan, “Sistem Purchase Order Otomatis pada UMKM,” J. Bisnis Teknol., vol. 6, no. 1, pp. 30–38, 2021. doi:10.31294/jbt.v6i1.2021.
[22]. A Putri, “Perbandingan Sistem Manual dan Digital untuk Manajemen Stok,” J. Teknol. UMKM, vol. 2, no. 1, pp. 20–28, 2020. doi:10.31227/osf.io/zf3ua.
[23]. G. Pramana, “Black Box Testing dalam Pengujian Aplikasi Web,” J. Rekayasa Sist., vol. 8, no. 1, pp. 55–63, 2021. doi:10.31227/osf.io/u4ycq.
[24]. J. H. Suharto, “Penerapan MVC Framework Laravel pada Aplikasi Retail,” J. Komput. Apl., vol. 10, no. 3, pp. 140–150, 2022. doi:10.31294/jka.v10i3.2022.
[25]. B. N. Putra, “Aplikasi Dashboard Inventori Berbasis Web,” J. Inf. Visual, vol. 4, no. 1, pp. 19–28, 2023. doi:10.31002/jiv.v4i1.2023.