Automatic Daily Scheduling System Using Cron Jobs for Handicraft Business Management

Lidia Terecia, Sandy Kosasi, David

STMIK Pontianak, Pontianak, Indonesia

Correspondence : *e-mail*: lidiatereciaa@gmail.com

Abstract

Ensuring accurate stock information remains a challenge for handicraft business managers when sales data from physical stores and e-commerce platforms are not updated simultaneously. Previous studies have addressed stock synchronization but have yet to employ scheduling mechanisms for delivering daily stock summaries integrated with sales data from both sources. This study developed an automated scheduling system using a cron job to record and summarize daily stock movements, utilizing Shopee API integration to ensure data consistency between the physical store and the online platform. The Reorder Point concept was applied to determine restocking thresholds, while the Twilio API was used to send stock summaries to managers via WhatsApp. The system was implemented using the Extreme Programming method to accelerate adaptation to user requirements. The results indicated that the system consistently provided daily stock reports, maintained data consistency across platforms, and supported procurement decision-making. This approach offers a novelty by combining automated stock synchronization and scheduled stock report distribution within a single integrated system.

Keywords*: Cron Job, Shopee API, Reorder Point, Twilio API, Extreme Programming.*

1. Introduction

Accurate inventory information is a cornerstone of operational reliability in omnichannel retail. Stock data that is not synchronized across platforms can undermine the integrity of supply chains and decision-making processes. Stock discrepancies between physical stores and e-commerce channels can result in customer dissatisfaction and elevated order cancellation rates [1]. This often disrupts the customer experience and directly affects merchant ratings on online marketplaces [2]. To prevent these issues, web-based stock management systems offer centralized access for inventory control while reducing the dependency on manual input [3].

However, not all solutions address the need for proactive restocking logic. This study proposes an integrated mechanism that unifies stock synchronization, low-stock alerts, and scheduled reporting into one system, offering a more comprehensive solution than previous approaches. With Shopee being one of the most widely used marketplaces in Indonesia, integrating its merchant API becomes essential to ensure stock consistency across platforms. [4]. API integration facilitates bidirectional data flow between the internal system and the external marketplace, preventing mismatches caused by asynchronous updates [5]. Furthermore, the system logs every inventory mutation, providing a detailed historical record of item inflow and outflow. [6].

To automate daily tasks, a cron job mechanism is implemented for generating scheduled stock summaries. These tasks run periodically without requiring human input, allowing business owners to receive timely reports [7]. Automated scheduling enhances consistency in reporting and minimizes the risk of delays in identifying critical inventory issues [8]. The system adopts the Reorder Point (ROP) method, a quantitative inventory control strategy that calculates when restocking should occur based on historical sales and lead time [9]. By incorporating ROP into the system logic, product availability can be maintained with greater accuracy and minimal manual oversight. Once stock levels reach a predefined minimum threshold, the system triggers an automated alert via the Twilio API, which delivers messages to managers or suppliers through WhatsApp [10]. This notification mechanism ensures timely procurement without relying on staff to constantly monitor inventory.

2. Research Method

This research adopts the Extreme Programming (XP) approach, which emphasizes iterative development and rapid responsiveness to user feedback. XP enables frequent delivery of working software, ensuring that user needs are addressed as early as possible in the development cycle [11]. The development process was divided into four core stages: planning, design, coding, and testing, each of which plays a crucial role in aligning the system with actual user workflows. [12]. This structured methodology fosters agility while maintaining system integrity across iterations. [13]. In the design phase, the Unified Modeling Language (UML) was employed to visualize the system architecture, component interactions, and user flows. Diagrams such as use cases, sequences, and activity charts were developed to guide implementation. [14]. To validate system reliability, white-box testing was performed using the basis path technique. This method ensures that every logical condition and execution path in the system is exercised and verified. [15].

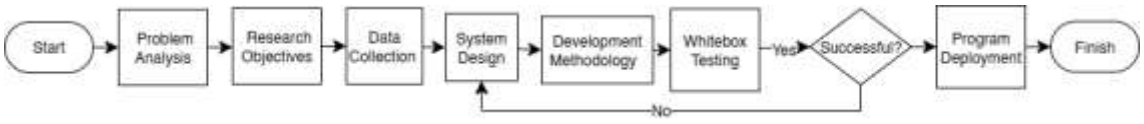


Figure 1. Research Flow

The system development process begins with problem identification, aiming to understand user needs and existing operational challenges. Subsequently, research objectives are clearly defined to guide data collection through methods such as observation, interviews, or literature review. The collected data supports system design, including UML diagrams, ERD construction, and database schema planning to ensure structural integrity. A suitable development methodology is then applied, followed by white-box testing; based on the results, the system is either deployed or returned to the design phase for refinement.

3. Results and Discussion

While many stock management systems emphasize synchronization and manual stock input, the present study introduces a scheduled automation mechanism that transcends traditional implementations. By leveraging the server-level cron job functionality, this system shifts from reactive stock monitoring to a proactive, time-based process that operates independently of user initiation. The cron job not only schedules report generation but also synchronizes stock mutation logs with reorder point evaluations, a layered interaction rarely seen in previous works. This approach provides a novel convergence of time-based automation, inventory analysis, and real-time notification, thereby transforming routine stock management into an anticipatory decision-support system. Based on the analysis and research objectives, the workflow and system interactions are illustrated in the use case diagrams presented in Figure 1.

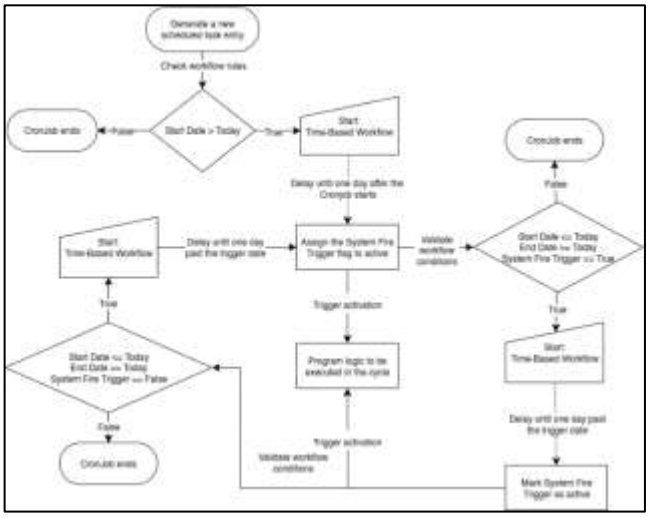


Figure 2(a) Cron Job Scheduling Flow

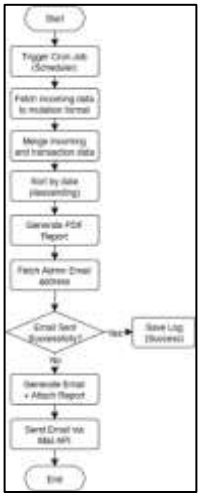


Figure 2(b) Automated Reporting Process.

Figure 2(a) shows the scheduling and triggering sequence of the cron job, which determines when the system initiates its operations and ensures that the process is executed only at predetermined times via specific triggers. This mechanism ensures that task execution occurs in a consistent and structured manner according to the defined cycle. Figure 2(b) illustrates the automated daily reporting process, which

commences once the cron job is activated, starting from data processing, followed by report generation, and concluding with delivery via email. This workflow guarantees that processed results are delivered promptly without requiring manual intervention, thereby linking both processes to maintain seamless system operation. Furthermore, the scheduled mechanism serves as a reference point for implementing the Reorder Point (ROP) calculation to determine the optimal timing for inventory replenishment. The ROP calculation is defined as follows:

$$ROP = (d \times L) + SS \quad (1)$$

In this equation:

- d = average daily sales
- L = lead time in days
- SS = safety stock

The Reorder Point method facilitates the determination of the optimal timing for placing replenishment orders based on the minimum safe inventory threshold. This calculation is derived from historical daily sales data, estimated supplier lead times, and predefined safety stock parameters. As illustrated in Figure 2, the ROP calculation is integrated with an API-based notification mechanism, which automatically transmits alerts to management when inventory levels reach the reorder threshold. Distinctively, this mechanism operates in tandem with a time-triggered automation process, wherein the ROP evaluation executes at scheduled intervals without manual intervention, ensuring that alerts correspond accurately to the latest inventory status recorded in the stock mutation log.

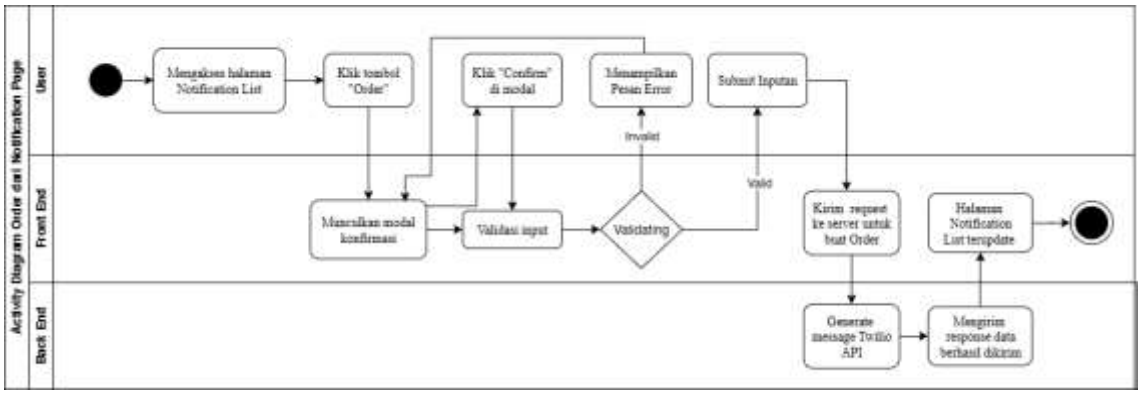


Figure 3. Flowchart of Twilio API Notification

As depicted in Figure 3, the process illustrates order placement via the Notification List page, which is integrated with the Twilio API. Once the user confirms the order and the data is validated, the system transmits a request to the server for processing. The back-end component subsequently invokes the Twilio API to dispatch a confirmation message directly to the supplier. Then it returns a success response to the front-end interface, thereby updating the order status. The automation of this sequence reduces delays typically associated with manual supplier communication. At the same time, the integration within a single notification interface ensures that message delivery, order verification, and stock threshold validation occur without navigating multiple system modules.

In addition, the system incorporates an automatic stock recap that delivers daily inventory summaries via WhatsApp, ensuring managers can monitor stock conditions without manual checking. A stock mutation log is also maintained to record detailed inbound and outbound transactions, which supports analysis of inventory movement and sales patterns. By combining notification, daily recap, and mutation history in one mechanism, this stage highlights the novelty of the research since earlier works only focused on synchronization or single-feature solutions. This comprehensive approach forms the basis for connecting the notification process with subsequent synchronization activities. As shown in Figure 3, the integration of the Shopee API enables automated synchronization of stock and order data between the internal system

and the Shopee platform, ensuring that both environments consistently reflect the same inventory and transaction information.

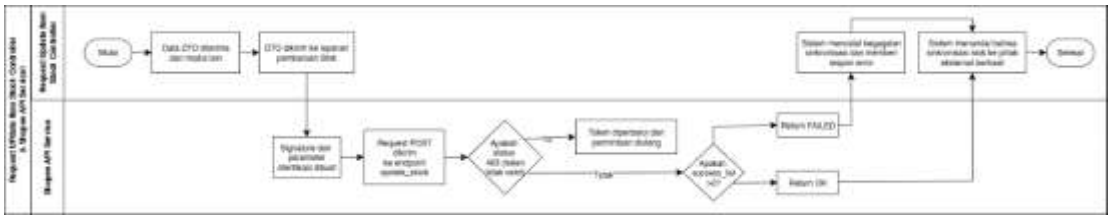


Figure 4. Use Case Diagram of Shopee API Integration

As shown in Figure 4, the Shopee API integration enables real-time stock updates as soon as incoming goods data are processed. This synchronization maintains data consistency between the internal system and the sales platform, thereby preventing discrepancies in stock quantities. The workflow also facilitates the seamless distribution of product information to the marketplace, ensuring that inventory management remains organized and easily monitored. Following the Shopee API integration process, the system records the receipt of incoming goods while simultaneously updating stock levels automatically, as demonstrated in Figure 5.

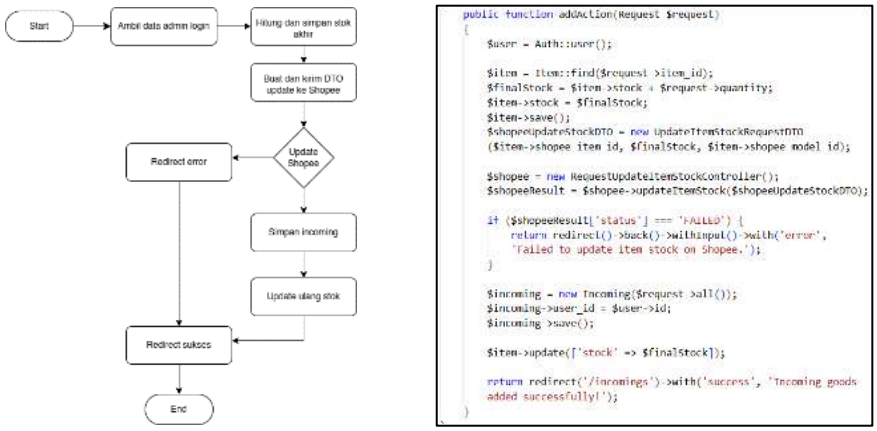


Figure 5. Pseudocode for Incoming Goods

As illustrated in Figure 5, the testing of the incoming goods pseudocode was conducted to ensure that the system is capable of accurately updating stock levels and recording item data. Utilizing the white-box testing approach with basis path testing, four independent execution paths were identified to evaluate various stock conditions, ranging from normal inventory levels to those meeting the criteria for both Safety Stock and the Reorder Point (ROP). This testing ensures that the generated reorder data is accurate and reliable. Subsequently, as shown in Figure 6, the system selects items whose stock levels are approaching the minimum threshold and displays them in the low-stock notification list.

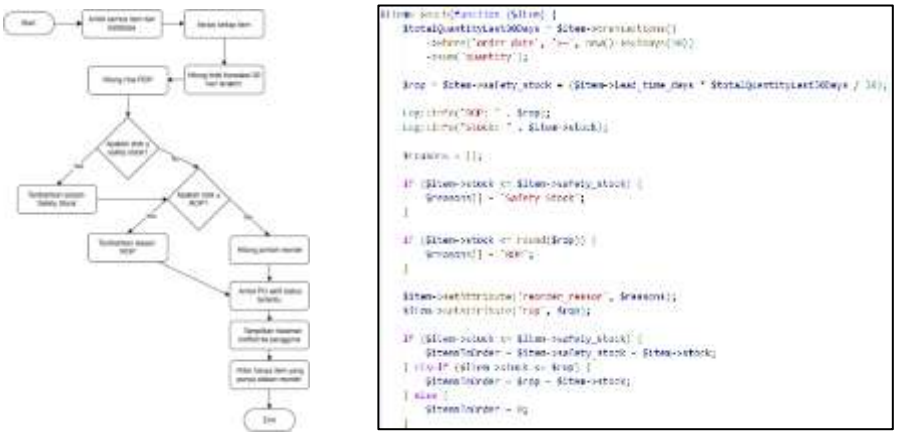
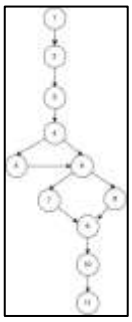
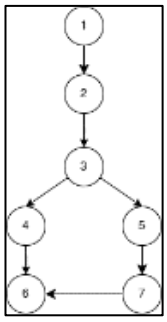


Figure 6. Pseudocode for Notification List

As depicted in Figure 6, the process begins with retrieving all item data from the database, followed by evaluating two stock conditions. Four distinct logical paths emerge based on the combinations of the conditions stock ≤ safety stock, stock≤safety stock, and stock ≤ ROP stock≤ROP, thereby ensuring that low-stock notifications are generated with precision. Subsequently, Figure 7 presents the middleware testing procedures, which verify that stock notifications are transmitted accurately and that stock updates within the system are executed according to the predefined conditions.



Gambar 7 (a) Notification Testing



Gambar 7 (b) Stock Update Testing

Figure 7(a) illustrates the middleware testing process for notifications within the index function of the NotiflistController, conducted using the white-box testing approach with basis path testing. The testing identified four independent execution paths corresponding to the scenarios of: normal stock levels, stock below the safety stock threshold, stock below the Reorder Point (ROP), and a combination of both conditions. The Cyclomatic Complexity value was calculated using the following formulas:

$$V(G) = E - N + 2 \text{ dan } V(G) = P + 1 \quad (2)$$

In this formula:

- E = the number of edges in the flow graph
- N = the number of nodes in the flow graph
- P = the number of predicates or logical decision points

Both formulas yielded a value of 4, representing the four logical paths tested. Figure 7(b) presents the middleware testing for stock updates, which comprises two independent execution paths: the first path occurs when the Shopee stock update is successful and the data are saved. In contrast, the second path occurs when the update fails, resulting in no data being stored. The calculation of both formulas produced a value of 2, corresponding to the two logical paths evaluated. Figure 8 illustrates the synchronization results between purchase orders and Shopee stock, where stock quantities are automatically adjusted following changes in the recorded data.

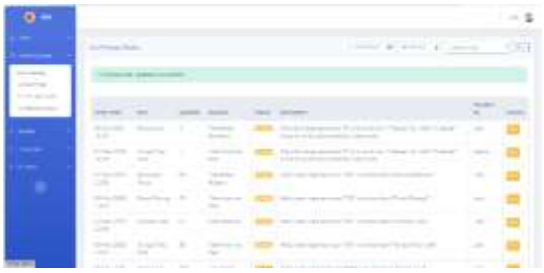


Figure 8 (a) Purchase Order List

Figure 8 (b) Synchronized Shopee Stock

Figure 8 (a) displays the purchase order list, which serves as the primary reference for adjusting product stock levels. Any modifications to this list automatically update the corresponding product stock quantities. Figure 8 (b) illustrates the Shopee stock, which is automatically adjusted to reflect these changes, whether they result from sales transactions or the receipt of incoming goods.

4. Conclusion

The daily automated scheduling system developed using a cron job successfully integrates Reorder Point (ROP) calculations, scheduled stock recording, cross-platform data synchronization, and notification delivery via the Twilio API into a single unified mechanism. The implementation of minimum stock

calculations, which take into account average daily sales and supplier lead times, enables alerts to be sent before inventory reaches critical thresholds. Integration with the Shopee API ensures that any stock changes in the system are immediately reflected on the marketplace, thereby maintaining stock consistency across all sales channels. Based on the implementation results, future development could focus on incorporating predictive sales trend analysis to support more optimal procurement planning.

References

[1] N. K. Akmal and M. N. Dasaprawira, "Rancang Bangun Application Programming Interface (API) Menggunakan Gaya Arsitektur GraphQL untuk Pembuatan Sistem Informasi Pendataan Anggota Unit Kegiatan Mahasiswa (UKM) Studi Kasus UKM Starlabs," *SITECH Journal*, vol. 5, no. 1, pp. 38–40, Jul. 2022. [Online]. Available: http://www.jurnal.umk.ac.id/sitech

[2] J. vom Brocke, A. Hevner, and A. Maedche, "Introduction to Design Science Research," in *Design Science Research. Lecture Notes in Information Systems and Organisation*, Cham: Springer, 2020, pp. 1–13. doi: 10.1007/978-3-030-46781-4_1.

[3] V. T. Gumilang, "Perancangan Sistem Manajemen Stok Barang Berbasis Web pada PT X," *Jurnal Ilmu Komputer dan Sistem Informasi*, 2023.

[4] C. Grob and A.-Schriftenreihe, "Inventory Management in Multi-Echelon Networks: On the Optimization of Reorder Points," [Online]. Available: http://www.autouni.de

[5] Juwita and F. Rahmiyatun, "Penerapan Metode Economic Order Quantity (EOQ) dan Reorder Point (ROP) pada Pengendalian Persediaan Bahan Baku di UMKM Dapur Bunga Berbintang," *Jurnal Maneksi*, vol. 12, no. 4, pp. 818–827, 2023.

[6] R. A. Bimantoro, A. S. Fitrani, and S. Busono, "Sistem WhatsApp sebagai Notifikasi pada UMSIDA Farm Store Berbasis Web," *Journal of Internet and Software Engineering*, vol. 1, no. 1, pp. 14–21, Jan. 2024, doi: 10.47134/pjise.v1i1.2248.

[7] R. S. Pressman and B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. New York: McGraw-Hill, 2014.

[8] S. M. Pratama and R. Jefri, "Analisis Sistem Akuntansi Persediaan Barang Dagang pada UMKM MiMa Frozen Food," *Jurnal Pundi*, vol. 8, no. 2, pp. 125–135, Dec. 2024, doi: 10.31575/jp.v8i2.565.

[9] T. F. Rahmadanti, M. Jajuli, and I. Purnamasari, "Klasifikasi Pengguna Shopee Berdasarkan Promosi Menggunakan Naïve Bayes," in *Proc. Seminar Nasional Teknologi Informasi dan Komunikasi (SENTIKA)*, Yogyakarta, Indonesia, 2021, pp. 81–89.

[10] N. Saprido and R. Pahlevi, "Pengembangan Aplikasi Manajemen Stock Barang pada PT Pangkalan Gas Andika Mukti Berbasis Web," *Bulletin of Informatics (BIN)*, vol. 2, no. 1, pp. 67–77, Jul. 2024, ISSN: 3025-7417. [Online]. Available: https://ojs.jurnalmahasiswa.com/ojs/index.php/bin.

[11] D. Fatmawati and D. A. Megawaty, "Aplikasi Supervisi Dosen Berbasis Web di Universitas XYZ," J. Informatika dan Rekayasa Perangkat Lunak (JATIKA), vol. 4, no. 3, pp. 270–283, 2023.

[12] A. D. Saputro and A. A. Narwastika, "Implementasi White Box Testing dengan Teknik Basis Path pada Pengujian Form Peminjaman Sistem Aplikasi Perpustakaan," in *Proc. Seminar Nasional Teknologi Informasi dan Bisnis (SENATIB)*, Surakarta, Indonesia, Jul. 2023, p. 406.

[13] D. Shenoy and R. Rosas, *Problems and Solutions in Inventory Management*. Cham: Springer International Publishing AG, 2018. doi: 10.1007/978-3-319-65696-0.

[14] S. Sundaramoorthy, *UML Diagramming: A Case Study Approach*. CRC Press, 2022.

[15] Safriza and H. Rohayani, "Implementation of Path Testing in White Box Testing for the National Public Transportation Driver Behavior and Competency Assessment System," *Lovelace Journal of Information System, Security, Education and Network Artificial Intelligence*, vol. 2, pp. 1–16, 2024.