Cloud-Native Inventory Management System Using Flutter and Appwrite for Inventory Control

Muhammad Fajar[1], Azwar[2], Baharuddin[3]
Informatics Study Program, Faculty of Computer Science, Universitas Ichsan Sidenreng Rappang

Effective inventory management is crucial for retail operations like Toko Karya Rappang, an office stationery and printing business. The shop faces challenges in managing 163 active products due to manual logbook recording, which causes data errors, delayed stock monitoring, reporting difficulties, and late stock replenishment. This research designs a cloud-native mobile inventory management system using Flutter and Appwrite as Backend-as-a-Service (BaaS). Developed via the Rapid Application Development (RAD) method for speed and flexibility, the system supports two user roles (owner and staff). Key features include incoming/outgoing goods tracking, real-time stock monitoring, low-stock notifications, and visual sales reports. System evaluation utilized White Box, Black Box, and User Acceptance Testing (UAT). White Box testing yielded low Cyclomatic Complexity scores (3 to 5), indicating minimal logical risk. Furthermore, the UAT involving 10 respondents achieved an 86.17% feasibility score, placing it in the "Good" category. The system is proven to function effectively, enhance accuracy, improve inventory efficiency, and serve as a viable reference for small and medium retail businesses.
Keywords: Inventory Management, Cloud-Native, Flutter, Appwrite, Rapid Application Development (RAD).

Corresponding Author:
Muhammad Fajar
Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Ichsan Sidenreng Rappang
muhammadfajar250723@gmail.com

1. Introduction

Karya Rappang Store is a retail business engaged in the sale of office stationery (ATK) and also provides services such as photo printing, passport photo photography, photo frame sales, document printing, and document photocopying [1]. The store is located in Lalebata District, Sidenreng Rappang Regency, South Sulawesi, and operates for extended hours, from 06:30 a.m. to 10:00 p.m. every day.

In its operational activities, the store employs eight female staff members who are assigned based on specific job functions. Some employees are responsible for the computer service section, assisting customers with photo printing, photography services, and document printing, while the remaining employees work in the stationery service section, handling stationery sales and document photocopying services [2]. This division of labor is implemented to ensure that services can be delivered more efficiently and systematically according to customer needs [3].

Despite its intensive operational activities, inventory recording at Karya Rappang Store is still carried out manually using notebooks [4]. This condition results in several challenges, including recording errors, delays in stock checking, and the absence of automated reporting. In addition, the lack of a notification system when stock levels are running low often causes delays in replenishment activities, which ultimately disrupt customer service operations [5].

Inventory management is a crucial aspect of store operations, including at Karya Rappang Store. Accurate and real-time inventory control is essential to prevent both stock shortages and overstock situations, which may lead to financial losses and decreased customer satisfaction [6].

Based on the study conducted by Saputri and Hirzan, the implementation of a Flutter- and Firebase-based mobile application for inventory management successfully reduced stock-checking time from 15 minutes to less than 1 minute and improved user satisfaction through an “excellent” UI/UX design [7]. Several other studies have also confirmed the advantages of Flutter in developing inventory management applications. For example, a mobile-based solution implemented in the meat business sector at UD. Puspa successfully addressed recording errors and improved the efficiency of real-time inventory management [8].

Furthermore, Flutter-based application development has been implemented across various domains, including point-of-sale (POS) and sales order systems. [9] designed a mobile POS application using Flutter and the Rapid Application Development (RAD) method, which was successfully validated through black-box testing with satisfactory results [10]. Widianto et al. also successfully developed a Flutter-based sales order application that achieved a user satisfaction level of 87.9% [11].

These findings indicate that Flutter is a highly suitable framework for the development of inventory management applications [12]. The framework offers high performance, responsive user interfaces on Android devices, and efficient cross-platform development capabilities [7][13]. In addition, Flutter can be easily integrated with real-time databases through backend platforms such as Appwrite [14].

However, significant challenges are still faced by Karya Rappang Store, where inventory management is conducted manually, making the process prone to errors, slow, and inefficient . Therefore, there is a need to develop a mobile application capable of automating inventory inflow and outflow recording, providing real-time inventory monitoring, and presenting sales reports that are easy for the store owner to understand.

Based on these problems and opportunities, this study is directed toward designing and developing a mobile application using Flutter specifically for inventory management needs at Karya Rappang Store [7], [13], [15], [16]. The development process utilizes the Rapid Application Development (RAD) method to ensure that the application can be developed quickly, flexibly, and in accordance with direct user feedback from the field. Accordingly, the proposed solution is expected to improve accuracy, efficiency, and user satisfaction in managing inventory and sales transactions, while simultaneously supporting store operations in a more professional and integrated manner.

Based on the literature review, a clear research gap can be identified. Previous studies have not adequately explored the implementation of a cloud-native inventory management system that simultaneously integrates real-time inventory control, automated low-stock notifications, sales reporting dashboards, multi-role user management, and Backend-as-a-Service (BaaS) architecture within a single mobile platform. Furthermore, studies focusing on inventory digitalization in small retail businesses with diverse product categories and service-oriented operations remain relatively limited. This gap highlights the need for a more comprehensive solution that addresses both operational and managerial requirements in retail inventory management.

The novelty of this study lies in the development of a cloud-native inventory management system that combines Flutter and Appwrite technologies to provide real-time inventory synchronization, automated stock threshold notifications, role-based access management for owners and staff, and interactive sales analytics within a single mobile application. Unlike previous studies that mainly focused on inventory recording or transaction processing, this research integrates inventory control, business monitoring, and decision-support functionalities into a unified platform tailored to the operational characteristics of Karya Rappang Store.

Accordingly, this study contributes both theoretically and practically. From a theoretical perspective, it extends the application of cloud-native architecture and mobile inventory management concepts within the

context of small retail enterprises. From a practical perspective, the proposed system provides an integrated solution for improving inventory accuracy, accelerating stock monitoring, reducing recording errors, supporting timely replenishment decisions, and enhancing managerial oversight through real-time sales reporting. These contributions distinguish the present study from previous research and demonstrate its potential as a reference model for inventory digitalization in similar retail businesses.

2. Methods

This study employed an applied research approach aimed at developing a mobile-based inventory management application to address inventory recording and monitoring problems at Karya Rappang Store. The system was developed using the Rapid Application Development (RAD) method, which emphasizes rapid system development, continuous user involvement, and iterative improvements. RAD was selected because it enables developers to respond quickly to user requirements while reducing development time and ensuring that the final system aligns with operational needs.

Data collection was conducted through direct observation and interviews with the store owner and employees. Observation was performed to understand existing inventory management procedures, while interviews were used to identify operational constraints, user requirements, and expectations regarding the proposed system. The developed application utilizes Flutter as the frontend framework, Appwrite as the Backend-as-a-Service (BaaS) platform, and Android Studio as the development environment.

Rapid Application Development (RAD) Stages

1. Requirements Planning
   The first stage focused on identifying business problems, system requirements, and user expectations. Observations and interviews revealed that inventory recording was still performed manually using notebooks, resulting in recording errors, delays in stock monitoring, and difficulties in generating reports. Based on these findings, the functional requirements of the system were established, including inventory recording, stock monitoring, low-stock notifications, sales reporting, and user management features.
2. User Design
   At this stage, system models and interface prototypes were designed collaboratively with prospective users. Database structures, use case diagrams, system workflows, and user interface layouts were developed to represent the operational processes of the store. User feedback was continuously incorporated to ensure that the design met actual business requirements. The design process included the development of role-based access mechanisms for owners and staff members.
3. Construction
   The construction phase involved the actual development of the application using Flutter and Appwrite technologies. Core functionalities implemented during this phase included product management, inventory transaction recording, stock monitoring, notification systems, sales reporting dashboards, and user authentication. Iterative testing and refinement were conducted throughout development to ensure that each feature functioned properly and aligned with user expectations.
4. Implementation
   The final stage involved deploying the system and evaluating its performance. The developed application was introduced to store owners and staff members for practical use. System evaluation was conducted using White Box Testing to assess program logic, Black Box Testing to verify functional requirements, and User Acceptance Testing (UAT) to measure user satisfaction and

system feasibility. The evaluation results were used to validate the effectiveness, usability, and reliability of the application.

Research Flowchart

The overall research process can be illustrated as follows:

1. Problem Identification
2. Observation and Interviews
3. Requirement Analysis
4. Requirements Planning (RAD)
5. User Design (Database, Interface, and System Design)
6. Application Development (Construction)
7. System Testing
   a. White Box Testing
   b. Black Box Testing
   c. User Acceptance Testing (UAT)
8. System Implementation
9. Evaluation and Improvement
10. Conclusion

3. Result And Discussion

Database Design

Table Relationships

The database design represents a structured data storage system in which data entities are interconnected and designed to support accurate, consistent, and efficient information management processes. In the Cloud-Native Inventory Management System using Flutter and Appwrite, the database is structured to ensure that inventory recording, inbound and outbound inventory transactions, product management, and user management processes can operate in an integrated manner. The relationships among the tables in this system illustrate the data flow utilized in the store's operational activities and can be seen in the following figure:

ISSN 2828-9463

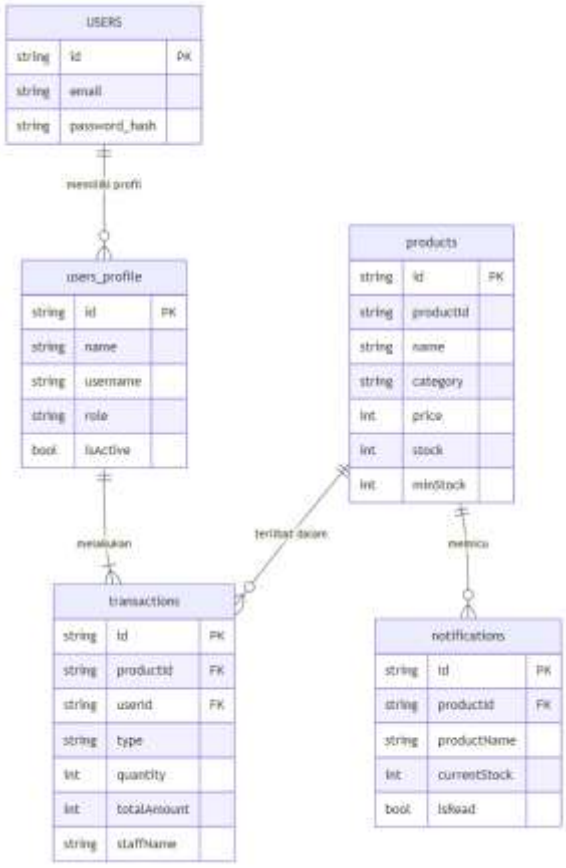

Figure 1. Database Table Relationships

USERS Database Table

Table 1. Users Database

Field	Data Type	Description
id	String	User ID
Email	String	User Email Address
Password	String	User Password

users_profile Database Table

Table 2. Users_Profile Database

Field	Data Type	Description
$id	String	User ID
userAuthId	String	Appwrite Authentication ID
name	String	User Name
role	Enum	User Role
isActive	Boolean	User Active Status
fcmToken[]	String	List of Firebase Cloud Messaging Tokens
$createdAt	Datetime	Date and Time When the User Profile Was Created
$updatedAt	Datetime	Date and Time When the User Profile Was Updated

Transactions Database Table

Table 3. Transactions Database

Field	Data Type	Description
$id	String	Unique ID

Field	Data Type	Description
transactionId	String	Transaction ID
productId	String	Product ID
userId	String	User ID (Owner/Staff)
productName	String	Product Name
quantity	Integer	Product Quantity
price	Integer	Product Unit Price
type	String	Transaction Type (Inbound/Outbound)
totalAmount	Integer	Total Transaction Value
category	String	Product Category
staffName	String	Staff Name
notes	String	Additional Notes
$createdAt	Datetime	Date and Time When the Transaction Was Created
$updatedAt	Datetime	Date and Time When the Transaction Was Updated

Products Database Table

Table 4. Products Database

Field	Data Type	Description
$id	String	Unique ID
productId	String	Product ID
name	String	Product Name
category	Enum	Product Category
price	Integer	Selling Price
stock	Integer	Current Stock Quantity
description	String	Product Description
minStock	Integer	Minimum Stock Threshold
unit	String	Unit of Measurement
$createdAt	Datetime	Date and Time When the Product Was Created
$updatedAt	Datetime	Date and Time When the Product Was Updated

notifications Database Table

Table 5. Notifications Database

Field	Data Type	Description
$id	String	Unique ID
notificationId	String	Notification ID
productId	String	Product ID
productName	String	Product Name
currentStock	Integer	Current Product Stock Quantity
minStock	Integer	Minimum Stock Threshold
message	String	Notification Message Content
isRead	Boolean	Notification Status
$createdAt	Datetime	Date and Time When the Notification Was Created
$updatedAt	Datetime	Date and Time When the Notification Was Updated

White Box Testing

Based on the results of the white box testing conducted on several core system functions, namely login authentication, product addition (add product), report calculation, and stock validation, it was found that all modules satisfied the program structural testing criteria. The testing was performed using the Cyclomatic Complexity method to measure the complexity of the program logic and to identify the independent paths that needed to be tested for each function [17].

The login authentication module obtained a Cyclomatic Complexity value of 2 with two independent paths, both of which were successfully executed without errors. Similarly, the add product module produced a Cyclomatic Complexity value of 2 with two independent paths that covered the entire product addition process under both successful and failed conditions. Furthermore, the report calculation module exhibited a higher level of complexity, with a Cyclomatic Complexity value of 5 and five independent paths covering all possible looping conditions, date validations, and inbound and outbound transaction types. Meanwhile, the stock validation module generated a Cyclomatic Complexity value of 4 with four independent paths that accommodated all conditions related to minimum stock checking, debounce mechanisms, and low-stock notification delivery [18].

All independent paths within each module were successfully executed without any logical errors, branching errors, or process flow failures. The obtained Cyclomatic Complexity values also indicate that the program complexity remains within a manageable and maintainable category. Therefore, it can be concluded that the program logic structure operates in accordance with the designed specifications, all core system functions perform correctly, and the white box testing of the inventory management application was successfully completed.

Black Box Testing

Based on the results of the functional testing conducted on the login, inventory transaction, product management, reporting, notification, user management, owner dashboard, and staff dashboard modules, it can be concluded that all features of the inventory management application operated in accordance with the specified functional requirements. Testing was performed through various scenarios, including both normal conditions and error-handling conditions, and all scenarios produced valid results with actual outputs matching the expected outputs. These findings demonstrate that the system possesses a high level of reliability in supporting user authentication, inventory management, transaction recording, report generation, user account management, and real-time inventory monitoring.

In addition, the system successfully implements effective data validation mechanisms to prevent input errors, including login validation, transaction quantity validation, stock validation, product data validation, and user data validation. The low-stock notification feature also proved to function effectively through both dashboard displays and push notifications, thereby assisting business owners in managing inventory more quickly and accurately. The support for real-time features, data filtering, report monitoring, and multi-role user management further enhances the effectiveness of the system in supporting business operations.

The developed application has successfully fulfilled all specified functional requirements and is suitable for implementation as an inventory management information system to support inventory control, transaction processing, reporting, and decision-making activities in an effective, efficient, and integrated manner.

User Acceptance Testing (UAT)

Table 6. Answer Options and Weighting Format for User Acceptance Testing

Code	Response	Weight
A	Very Easy/Good/Appropriate/Clear	5

Code	Response	Weight
B	Easy/Good/Appropriate/Clear	4
C	Neutral	3
D	Fairly Difficult/Fairly Good/Fairly Appropriate/Fairly Clear	2
E	Very Difficult/Poor/Inappropriate/Unclear	1

Table 7. Questionnaire Format for User Acceptance Testing

Code	Question	A	B	C	D	E
P1	Does the login process run smoothly and quickly using the provided account credentials?	?	?	?	?	?
P2	Is the inventory stock information displayed on the dashboard accurate and updated in real time?	?	?	?	?	?
P3	Is the inventory stock information displayed on the dashboard accurate and updated in real time?	?	?	?	?	?
P4	Does the product search feature make it easier for you to find the desired products?	?	?	?	?	?
P5	Is the process of recording incoming stock (restocking) and outgoing stock (sales transactions) easy for staff to perform?	?	?	?	?	?
P6	Does the system successfully provide automatic notifications or warnings when stock levels are low?	?	?	?	?	?
P7	(Owner Only) Does the user management feature (adding/deactivating staff accounts) function as expected?	?	?	?	?	?
P8	Is the application's interface (colors, icons, and text) visually appealing and easy to understand?	?	?	?	?	?
P9	Is the application’s menu layout and navigation easy for new users to learn?	?	?	?	?	?
P10	Does the application perform responsively (without noticeable delays) when navigating between pages?	?	?	?	?	?
P11	Is using this application more efficient than the previous manual inventory recording system?	?	?	?	?	?
P12	Does this application help minimize inventory recording errors in the store?	?	?	?	?	?

Table 8. Questionnaire Response Results for User Acceptance Testing

Code	Answer					Percentage (%)				
	A	B	C	D	E	A	B	C	D	E
P1	10	0	0	0	0	100	0	0	0	0
P2	10	0	0	0	0	100	0	0	0	0
P3	2	7	1	0	0	20	70	10	0	0
P4	2	7	1	0	0	20	70	10	0	0
P5	7	0	3	0	0	70	0	30	0	0
P6	10	0	0	0	0	100	0	0	0	0
P7	0	2	0	0	0	0	20	0	0	0
P8	5	5	0	0	0	50	50	0	0	0
P9	4	6	0	0	0	40	60	0	0	0
P10	6	4	0	0	0	60	40	0	0	0

Code	Answer					Percentage (%)				
	A	B	C	D	E	A	B	C	D	E
P11	7	2	1	0	0	70	20	10	0	0
P12	6	4	0	0	0	60	40	0	0	0

Number of Respondents:

2 Users (Owners)

8 Users (Staff Members)

Table 9. User Acceptance Testing Results

Code	Score (Number of Responses × Weight)					TTL	TTL/USER	%
	A	B	C	D	E			
P1	50	0	0	0	0	50	5	100
P2	50	0	0	0	0	50	5	100
P3	10	28	3	0	0	41	4.1	82
P4	10	28	3	0	0	41	4.1	82
P5	35	0	9	0	0	44	4.4	88
P6	50	0	0	0	0	50	5	100
P7	0	8	0	0	0	8	0.8	16
P8	25	20	0	0	0	45	4.5	90
P9	20	24	0	0	0	44	4.4	88
P10	30	16	0	0	0	46	4.6	92
P11	35	8	3	0	0	46	4.6	92
P12	30	16	0	0	0	46	4.6	92

Notes :

(% = (Total / User) / 5) * 100)

% >= 90% A

80 <= % <= 89 B

70 <= % <= 79 C

60 <= % <= 69 D

50 <= % <= 59 E -> The system is not suitable, it needs to be fixed regarding the question.

Final Score (Average) = Total % / 10

(P1+P2+P3+P4+P5+P6+P7+P8+P9+P10+P11+P12) / 10 = 100

Value Category:

% >= 90% A

80 <= % <= 89.99 B

70 <= % <= 79.99 C

60 <= % <= 69.99 D

50 <= % <= 59.99 E

So the final result of the User Acceptance Testing is 86.17% with a value category of

Discussion of the Model

a. Login Page

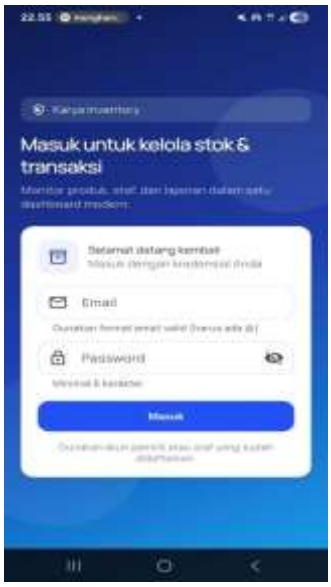

Figure 2. Login Interface

The system interface begins with the Login page, which serves as the primary security gateway of the application. On this page, users are required to enter their registered credentials, consisting of an email address and password. The system then validates the data through the Appwrite server to verify the user's identity and automatically detect the user role, whether as an Owner or Staff member. This mechanism ensures that each user is directed only to the dashboard page corresponding to their access rights, thereby maintaining the security of the store's sensitive data.

b. Owner Dashboard Page

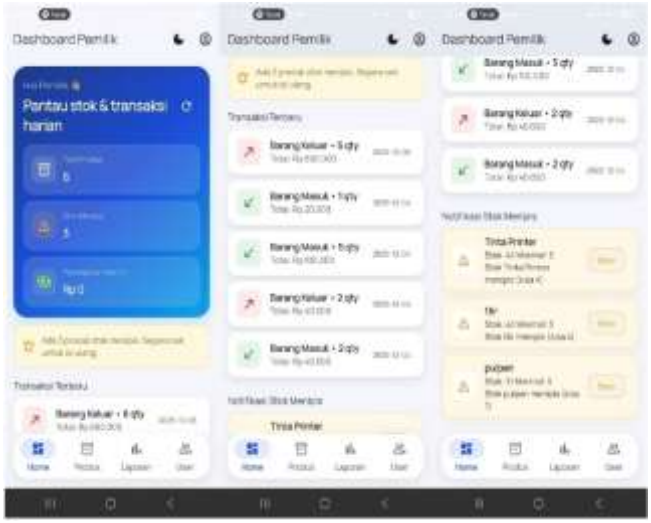

Figure 3. Owner Dashboard Interface

After successfully logging in, the store owner is presented with the Owner Dashboard page. This interface is designed as an executive information center that provides a real-time summary of the store's key performance indicators. Through this dashboard, the owner can monitor essential information such as the total number of registered products, daily revenue summaries, and low-stock warning notifications. This feature is particularly important in assisting the owner in making prompt procurement decisions without the need to physically inspect the warehouse.

c. Product and Product Addition Page for Owners

Figure 4. Product and Add Product Interface

After successfully logging in, the store owner is presented with the Owner Dashboard page. This interface is designed as an executive information center that provides a real-time summary of the store's key performance indicators. Through this dashboard, the owner can monitor essential information such as the total number of registered products, daily revenue summaries, and low-stock warning notifications. This feature is particularly important in assisting the owner in making prompt procurement decisions without the need to physically inspect the warehouse.

d. Owner Report Page

Figure 5. Owner Report Interface

To support business analysis needs, the system provides a Report Page that presents financial recapitulations in a transparent manner. Through this section, the owner can review the store's cash flow, including total revenue (gross income), expenses related to inventory replenishment (restocking), and daily gross profit calculations. This information is provided to facilitate the owner in evaluating monthly profitability and tracking transaction histories within a specified period.

ISSN 2828-9463

e. Owner Staff Management Page

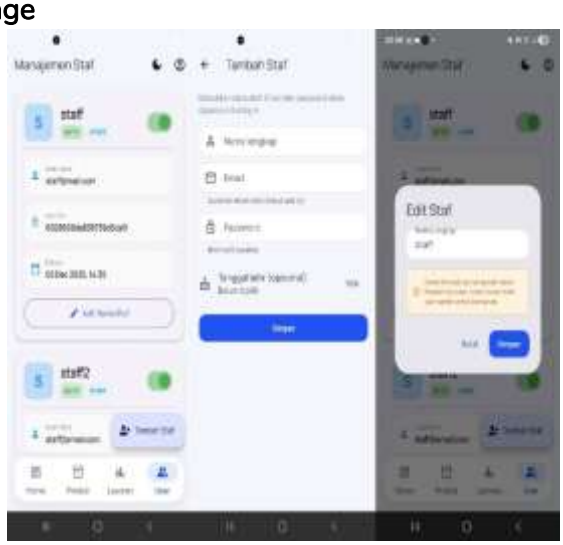

Figure 6. Owner Staff Management Interface

This feature is specifically designed to provide management authority to the owner in managing human resources. Through this page, the owner can register new staff accounts, update user information, or deactivate access for employees who are no longer working at the store. This configuration ensures that only active employees have access to the store's operational system, thereby maintaining the integrity of transaction data.

f. Owner User Profile Page

Figure 7. Owner User Profile Interface

The user profile section displays the personal account information of the currently logged-in owner, including the user's name and email address. In addition to serving as an identity display, the profile page also provides a logout button, ensuring that user sessions can be securely terminated.

g. Staff Dashboard Page

Figure 8. Staff Dashboard Interface

Unlike the owner dashboard, the Staff Dashboard is specifically focused on improving daily operational efficiency. This page provides staff members with quick access to inventory availability information. Stock information is displayed clearly, enabling staff to respond promptly to customer inquiries regarding product availability without delays.

h. Staff Transaction Input Page

Figure 9. Staff Transaction Input Interface

This page serves as the primary working interface for staff members, where inventory inflow (restocking) and outflow (sales) transactions are recorded. The interface is designed to be simple yet functional, allowing staff to select transaction types, search for product names, and enter item quantities. Every transaction entered automatically updates stock quantities in the central database, thereby minimizing manual calculation errors.

i. Staff History Page

Figure 10. Staff History Interface

To ensure accountability, the system provides a history page that records staff activity logs. This list displays transaction records processed by the staff account on that particular day, including details of products entered into or removed from inventory. This feature facilitates audit trail activities in the event of discrepancies between physical inventory and digital records.

j. Staff Profile Page

Figure 11. Staff Profile Interface

Completing the staff interface series, this profile page displays the identity of the currently active staff user. Similar to the owner profile page, this section functions as a basic account management interface and provides access to log out of the system after working hours.

Maintenance

The maintenance phase is carried out after the Cloud-Native Inventory Management System (Flutter and Appwrite) has been successfully implemented and put into operational use at Karya Rappang Store. The purpose of maintenance is to ensure system stability, prevent errors (bugs) in inventory synchronization processes, and guarantee that the application can adapt to technological updates and future changes in store management requirements.

4. Conclusion

Based on the research findings, the author draws the following conclusions:

1. Successful System Implementation: This study successfully developed a Cloud-Native Inventory Management System by utilizing Flutter as the frontend technology and Appwrite as the backend platform. The system has proven effective in replacing the conventional inventory recording method previously implemented at Karya Rappang Store, thereby addressing issues related to data inaccuracies and the slow process of inventory recapitulation.
2. Effectiveness of Monitoring and Notification Features: The developed system is capable of providing real-time information regarding product availability. The minimum stock notification feature functions effectively by delivering early warnings to users, which directly contributes to preventing stockouts that could potentially disrupt sales operations.
3. System Validity and User Acceptance: Based on the results of Black Box Testing, all system functionalities, including product management, inbound and outbound inventory transactions, and user management, operated in accordance with the designed logic without any functional errors. Furthermore, the results of the User Acceptance Test (UAT) indicate a very high level of user acceptance among both store owners and staff. This positive evaluation was reflected in terms of usability, interface quality, and the relevance of system features to the operational needs of the store.

Despite the positive outcomes, several limitations should be acknowledged. First, the system was developed and evaluated within a single retail business environment, namely Karya Rappang Store, which may limit the generalizability of the findings to other business sectors with different operational characteristics. Second, the evaluation focused primarily on functional performance and user acceptance, while long-term impacts on business productivity, inventory cost reduction, and financial performance were not assessed. Third, the application currently relies on manual transaction input by users and does not yet integrate with barcode scanning, supplier management modules, or advanced inventory forecasting mechanisms.

Recommendations for Future Research

Future studies are recommended to expand the implementation of the system across multiple retail businesses to evaluate scalability and adaptability in different operational contexts. Further development may also incorporate barcode or QR code scanning technology to improve transaction accuracy and efficiency. In addition, integrating supplier management, purchase order automation, and inventory forecasting based on historical sales data could enhance the decision-support capabilities of the system. The application may also be extended through the implementation of artificial intelligence and machine learning techniques to predict inventory demand, optimize stock levels, and support strategic business planning. Finally, future research should conduct longitudinal evaluations to measure the long-term effects of system adoption on operational efficiency, inventory costs, profitability, and organizational performance.

5. References

[1] N. S. Fadillah And J. Sutopo, “Implementasi Metode Fifo Pada Sistem Informasi Dalam Mengelola Persediaan Barang Berbasis Web,” *Jurnal Riset Dan Aplikasi Mahasiswa Informatika (Jrami)*, Vol. 5, No. 2, Pp. 357–366, Apr. 2024, Doi: 10.30998/Jrami.V5i2.10579.

[2] F. P. J. Sibuea, D. Agustin, A. Ferdhinand, W. Widyatmoko, D. Nomensen, And A. Kusmawati, “Rancang Bangun Sistem Inventory Barang Berbasis Web Dengan Metode Prototyping Di Program Studi Teknologi Rekayasa Otomotif Politeknik Stmi Jakarta,” *Ilkomnika: Journal Of Computer Science And Applied Informatics*, Vol. 6, No. 1, Pp. 91–101, Apr. 2024, Doi: 10.28926/Ilkomnika.V6i1.608.

[3] A. Y. Hasan, Y. N. Dewi, A. S. A. Budiyono, And R. I. Setiawan, "Implementasi Sistem Informasi Inventory Pada Momo Coffee," *Bit-Tech*, Vol. 7, No. 2, Pp. 515–524, Dec. 2024, Doi: 10.32877/Bt.V7i2.1885.

[4] H. J. Zietsman And J. H. Van Vuuren, "A Generic Decision Support Framework For Inventory Procurement Planning In Distribution Centres," *Comput. Ind. Eng.*, Vol. 177, P. 109057, Mar. 2023, Doi: 10.1016/J.Cie.2023.109057.

[5] Ahmad Zainudin, Andik Prakasa Hadi, And Agus Priyadi, "Sistem Informasi Persediaan Obatberbasis Web Di Rumah Sakit Bina Kasih," *Jurnal Ilmiah Sistem Informasi*, Vol. 3, No. 3, Pp. 30–34, Jan. 2025, Doi: 10.51903/2xwvpm83.

[6] A. Saputri And A. M. Hirzan, "Aplikasi Manajemen Inventori Berbasis Mobile Menggunakan Flutter Dan Firebase Realtime Database," *Jurnal Informatika Dan Teknik Elektro Terapan*, Vol. 12, No. 3, Pp. 1586–1592, 2024, Doi: 10.23960/Jitet.V12i3.4324.

[7] A. Saputri And A. M. Hirzan, "Aplikasi Manajemen Inventori Berbasis Mobile Menggunakan Flutter Dan Firebase Realtime Database," *Jurnal Informatika Dan Teknik Elektro Terapan*, Vol. 12, No. 3, Pp. 1586–1592, 2024, Doi: 10.23960/Jitet.V12i3.4324.

[8] I. M. B. T. Dewangga, K. D. P. S. Putra, N. N. Supuwiningsih, And N. Made D. K. Putri, "Perancangan Aplikasi Pengelolaan Data Inventory Daging Berbasis Mobile Menggunakan Framework Flutter (Studi Kasus : Ud. Puspa)," *Prosiding Seminar Hasil Penelitian Informatika Dan Komputer 2025*, Vol. 2, No. 1, P. 2025, 2025.

[9] S. Tjandra And G. S. Chandra, "Pemanfaatan Flutter Dan Electron Framework Pada Aplikasi Inventori Dan Pengaturan Pengiriman Barang," *Journal Of Information System,Graphics, Hospitality And Technology*, Vol. 2, No. 02, Pp. 76–81, Dec. 2020, Doi: 10.37823/Insight.V2i02.109.

[10] M. A. Sumarto, "Analisis Dan Perancangan Aplikasi Point Of Sale (Pos) Untuk Usaha Mikro, Kecil, Dan Menengah (Umkm) Dengan Metode Rapid Application Development (Rad)," *Jurnal Studi Komunikasi Dan Media*, Vol. 27, No. 1, Pp. 17–34, 2023, Doi: 10.17933/Jskm.2023.5115.

[11] And Y. Y. Widianto, Y. H. Pesik, "Perancangan Dan Pengembangan Aplikasi Mobile Sales Order Berbasis Flutter Pada Perusahaan Dua Jaya," *Jurnal Mahasiswa Teknik Informatika*, Vol. 8, P. 5, 2024.

[12] M. B. Khanyi, S. N. Xaba, N. A. Mlotshwa, B. Thango, And L. Matshaka, "A Roadmap To Systematic Review: Evaluating The Role Of Data Networks And Application Programming Interfaces In Enhancing Operational Efficiency In Small And Medium Enterprises," *Sustainability*, Vol. 16, No. 23, P. 10192, Nov. 2024, Doi: 10.3390/Su162310192.

[13] M. A. Sumarto, "Analisis Dan Perancangan Aplikasi Point Of Sale (Pos) Untuk Usaha Mikro, Kecil, Dan Menengah (Umkm) Dengan Metode Rapid Application Development (Rad)," *Jurnal Studi Komunikasi Dan Media*, Vol. 27, No. 1, Pp. 17–34, 2023, Doi: 10.17933/Jskm.2023.5115.

[14] A. A. U. Nuha, M. Migunani, M. U. Dewi, K. Rozikin, And A. A. Kuncoro, "Rapid Application Development Of A Mobile Stock Management System," *Jurnal Ilmiah Sistem Informasi*, Vol. 4, No. 3, Pp. 708–723, Nov. 2025, Doi: 10.51903/Rzrga338.

[15] I. M. B. T. Dewangga, K. D. P. S. Putra, N. N. Supuwiningsih, And N. Made D. K. Putri, "Perancangan Aplikasi Pengelolaan Data Inventory Daging Berbasis Mobile Menggunakan Framework Flutter (Studi Kasus : Ud. Puspa)," *Prosiding Seminar Hasil Penelitian Informatika Dan Komputer 2025*, Vol. 2, No. 1, P. 2025, 2025.

[16] And Y. Y. Widianto, Y. H. Pesik, "Perancangan Dan Pengembangan Aplikasi Mobile Sales Order Berbasis Flutter Pada Perusahaan Dua Jaya," *Jurnal Mahasiswa Teknik Informatika*, Vol. 8, P. 5, 2024.

[17] B. Sudradjat, "Penggunaan Teknologi Flutter Dalam Aplikasi Mobile Untuk Pengembangan Kedai Kopi," *Remik*, Vol. 6, No. 1, Pp. 1–8, Oct. 2021, Doi: 10.33395/Remik.V6i1.11123.

[18] D. K. Hiuredhy And Y. R. Beeh, "Aplikasi Reservasi Ibadah Mawar Sharon Salatiga Menggunakan Flutter," *Jatisi (Jurnal Teknik Informatika Dan Sistem Informasi)*, Vol. 9, No. 3, Pp. 1739–1751, Sep. 2022, Doi: 10.35957/Jatisi.V9i3.2161.