Eduvest – Journal of Universal Studies
Volume 6 Number 9, September, 2026
p- ISSN 2775-3735- e-ISSN 2775-3727

Design and Evaluation of a Flutter Node.js Mongo DB System for Hospital Nutritional Inventory Auditing

Raihan Muflih*, Joko Aryanto
Universitas Teknologi Yogyakarta, Indonesia
Email: raihanmuflihjurnal@gmail.com*

Keywords
Distributed System; Firebase; Flutter; Hospital Logistics; MongoDB; Node.js;

ABSTRACT
Manual auditing of hospital nutritional supplies often results in delays, transcription errors, and discrepancies between physical stock and system records. This study designed and evaluated a mobile-cloud inventory auditing system for the nutritional installation at RSUD Kraton Pekalongan. The system was developed using the Waterfall Software Development Life Cycle (SDLC) model and implemented with a Flutter mobile application, a Node.js backend, a MongoDB database, and Firebase-based data synchronization. Actual operational data from a one-month deployment in April 2025 were used to evaluate the system against the previous manual workflow. The results showed an input accuracy of 98.47% (1,226 valid entries out of 1,245), a 90% reduction in inventory discrepancies, and an 80% reduction in auditing time, from 4.5 hours to 0.9 hours per section. Furthermore, network latency evaluation showed an average optimized payload ingestion time of 0.8 seconds under stable connectivity, supported by a low database input/output (I/O) footprint of 310.91 B/s for inbound traffic. These findings indicate that the proposed system substantially improved data consistency, reduced the burden of manual transcription, and accelerated hospital logistics auditing without requiring excessive network bandwidth. The study contributes a validated implementation model for mobile-cloud inventory management in healthcare settings and provides a practical foundation for future predictive inventory planning.

INTRODUCTION

Clinical dietary logistics require a high degree of precision and timely data availability to fulfill patient nutritional requirements and support continuous hospital operations. Healthcare inventory management differs substantially from standard commercial retail because of the critical nature of the commodities involved. Medical dietary supplies directly affect patient recovery, the effectiveness of dietary interventions, and overall clinical outcomes. Discrepancies within this highly sensitive supply chain may compromise patient care, increase operational costs, and disrupt procurement forecasting. Conventional inventory management in many regional healthcare facilities still relies heavily on manual auditing methods. Staff at RSUD Kraton Pekalongan, for example, reported significant inefficiencies caused by repetitive manual transcription activities. Personnel were frequently required to perform redundant data entry, initially recording stock information on paper cards before transferring it into local

desktop systems. This manual transition introduces human error, typographical mistakes, and synchronization delays that can hinder timely operational decision-making.

Addressing these systemic bottlenecks requires a transition from localized manual recording to an interconnected digital ecosystem. Previous research has shown that manual inventory management processes, in which incoming and outgoing goods are recorded without integrated digital systems, frequently result in discrepancies between inventory records and actual stock conditions (Suprianto, 2024; Akmal et al., 2026). The healthcare sector has increasingly recognized the need to digitize its supply chain; however, the implementation of such systems often faces challenges related to usability, network stability, and data governance (Nasiri Khoshroudi et al., 2025). Historically, hospital information systems (HIS) have often relied on monolithic architectures and relational database management systems (RDBMSs). Although RDBMSs are well suited to structured transactional data and support ACID (Atomicity, Consistency, Isolation, and Durability) properties, rigid schemas may be less adaptable when inventory data structures evolve rapidly or contain heterogeneous attributes (Ming & Ariffin, 2025; Abbasi et al., 2026).

The transition to document-oriented NoSQL databases, such as MongoDB, can provide greater schema flexibility for clinical inventory data than conventional relational approaches. This flexibility can support the incorporation of new inventory attributes or dietary categories without requiring extensive schema redesign (Sen & Mukherjee, 2024). Furthermore, integrating cross-platform mobile frameworks can reduce repetitive manual recording by enabling direct data entry and synchronization between mobile devices and centralized inventory systems (Feng et al., 2026). Flutter, an open-source user interface (UI) software development kit, supports application development from a shared codebase across multiple platforms, thereby reducing development overhead while maintaining near-native performance (Zhang, 2026).

In conjunction with front-end development, full-stack JavaScript architectures using Node.js and Express can streamline backend processing for healthcare applications (Sharma, 2018; Lestari et al., 2026). Node.js employs an asynchronous, event-driven architecture that is well suited to handling multiple concurrent input/output operations. This characteristic is particularly relevant in hospital environments where several logistics staff members may simultaneously audit inventory across different storage areas (Nadim et al., 2024).

Despite these technological developments, many contemporary studies continue to focus primarily on architectural descriptions and theoretical frameworks. Limited evidence is available regarding their direct effects on input accuracy and auditing time in real-world hospital stock opname scenarios involving offline-to-online synchronization (Han et al., 2026; Zulfikar & Putra, 2025). Furthermore, empirical evaluations of cross-platform applications under specific healthcare constraints, such as fluctuating intranet bandwidth, connectivity dead zones in storage facilities, and stringent data governance requirements, remain limited (Sarabu, 2024; Oktaviyanti & Haryono, 2025). A research gap therefore persists regarding the integrated use of mobile data entry, synchronization services, and backend validation for hospital nutritional stock opname (Baig et al., 2026).

This study addressed that gap by developing and evaluating an end-to-end integrated system in a real-world hospital setting. Specifically, the study aimed to design and evaluate a mobile-cloud inventory auditing system based on a cross-platform technology stack and assess

\

its performance using operational and technical metrics (Pratrian et al., 2026; Dwipanilih et al., 2025). The implementation was intended to improve operational visibility, reduce manual auditing time, strengthen data consistency across multiple hospital storage areas, and support more reliable nutritional inventory management. By reporting latency, scalability, and network payload metrics, the study provides a technical reference for regional hospitals seeking to modernize inventory auditing processes through distributed digital architectures.

The primary contribution of this research is the development and evaluation of a mobile-cloud solution designed to improve inventory data accuracy and auditing efficiency in a hospital setting. The study also provides empirical evidence regarding the applicability of Flutter, Node.js, MongoDB, and related synchronization technologies for healthcare logistics under constrained network conditions. In addition, it offers a practical implementation reference for regional hospitals seeking to transition from paper-based inventory processes to more integrated digital inventory management systems without requiring extensive infrastructure expansion.

To support the development of a resilient and scalable hospital inventory auditing system, the study drew on recent advances in software engineering, database management, and mobile-cloud architecture. The following literature review positions the study within the broader context of healthcare informatics and digital inventory management.

The Shift to Non-Relational Databases in Clinical Settings

The rigid schema structure of traditional SQL databases may create limitations in clinical environments where inventory categories and data attributes change frequently. Recent studies have shown that document-oriented NoSQL databases, such as MongoDB, can provide greater flexibility for certain healthcare applications (Khan et al., 2024). MongoDB stores data in BSON (Binary JSON) format, allowing documents to contain heterogeneous structures. This flexibility can be useful in hospital dietary logistics, where the attributes of perishable goods, such as expiration dates and temperature requirements, may differ substantially from those of nonperishable supplies. According to a 2025 study on healthcare data warehousing, NoSQL integration reduced query latency by up to 40% when managing unstructured medical inventory data compared with SQL-based alternatives, thereby supporting higher-throughput data ingestion (Guntupalli, n.d.)

Cross-Platform Mobile Engineering in Healthcare

Developing separate native mobile applications for Android and iOS can require substantial development resources, which may be difficult for regional hospitals with limited information technology budgets. Cross-platform frameworks such as Flutter provide a shared-codebase approach that can reduce development and maintenance effort. Flutter uses the Dart programming language and supports compilation to native machine code on supported platforms (Ambati, 2025). Studies evaluating cross-platform tools for enterprise data collection have highlighted Flutter's ability to support complex state management and background synchronization, which may be useful for hospital applications operating under intermittent Wi-Fi connectivity (Guntupalli, n.d.).

Distributed Cloud Synchronization and Firebase

In hospital inventory auditing, maintaining data availability during temporary network disruptions is important. Architectures that depend entirely on continuous client-to-server connectivity may be vulnerable to interruptions. To address this issue, distributed systems may

incorporate synchronization and local persistence mechanisms. Firebase services can support real-time synchronization, authentication, and local data persistence. Studies of synchronization under intermittent connectivity indicate that Firebase can cache data locally and synchronize updates after connectivity is restored (Setiaman, 2025). This offline-capable approach supports inventory recording in storage areas where network connectivity may be unstable.

SDLC Waterfall in Regulated Healthcare IT

Although Agile and Scrum methodologies are widely used in software development, structured sequential development models such as the Waterfall Software Development Life Cycle (SDLC) may still be applied in regulated healthcare information technology projects where documentation, validation, and traceability are important (Hermawan et al., 2025). The sequential structure of the Waterfall model can support systematic requirements analysis, architecture design, implementation, testing, and documentation before deployment (Zhang et al., 2024). In this study, the Waterfall SDLC was used to provide a clearly documented development process and to support verification of system requirements and data governance considerations at each stage (Ștefan et al., 2024).

METHODS

This study designed and evaluated an integrated web- and mobile-based application architecture for hospital inventory stock opname. The proposed system provided a synchronized digital framework to replace paper-based auditing processes, improve data availability, and reduce repetitive manual workload (Wei et al., 2026). The methodology included the software development process, user workflows, empirical observation, system evaluation metrics, and load testing.

Research Design and SDLC Implementation

This research adopted the Software Development Life Cycle (SDLC) Waterfall model to develop and evaluate a hospital inventory auditing system. The Waterfall methodology was selected because the requirements assessment, design, implementation, and validation workflows in a healthcare environment necessitate stringent documentation, predictable milestones, and structured approvals before deployment to comply with institutional standards. The methodology progressed through five sequential phases:

Requirements AnalysisIn-depth: interviews and observational studies were conducted with the nutritional staff at RSUD Kraton Pekalongan to map the existing pain points in the manual ledger system, identifying high-priority needs such as native barcode scanning, offline-mode capabilities, and real-time variance alerts.

System Design: The architectural topology was mapped, selecting Flutter for the mobile interface, Node.js for the API middleware, MongoDB for document storage, and Firebase for authentication and real-time state management.

Implementation: The codebase was constructed. Flutter widgets were integrated with device camera APIs for barcode scanning. The Node.js Express server was configured to handle RESTful endpoints, while MongoDB schemas were dynamically structured using Mongoose Object Data Modeling (ODM).

Testing: Black-box functional testing, JMeter load testing, and network latency evaluations were executed to ensure system stability under simulated hospital network

\

constraints. Deployment and Maintenance: The system was officially deployed for a one-month operational test in April 2025 to gather authentic empirical validation data from the field.

User Actor Identification and Authorization Constraints

To ensure data integrity and structural transparency within the distributed framework, role-based access control (RBAC) was implemented. Three primary user actors were identified within the hospital ecosystem, each possessing specific operational boundaries to prevent unauthorized data manipulation:

Field Logistics Staff (Mobile Client User): Responsible for executing the physical stock counting across the dietary storage wards. This user interacts exclusively with the Flutter mobile application to perform barcode scanning, verify commodity counts, and update localized physical quantities |Q_physical |. They are restricted from altering historical data or modifying pricing variables.

Nutrition Installation Data Analyst (Web Dashboard Administrator): Responsible for auditing, reconciliation, and administrative validation of data logs. This actor utilizes the centralized web interface to crosscheck automatically flagged variances, generate inventory mutation records, process inbound vendor deliveries, and manage procurement invoice transactions.

Head of Nutrition Installation (Management/Supervisor): Acts as the high-level approver who monitors the analytical metrics, validates macro-inventory balances, and oversees operational efficiency reports. This role possesses read-only access to strategic data visualizations and audit summaries.

Data Collection Protocol and Empirical Authenticity

Empirical validation was performed utilizing authentic dataset constraints gathered during an intensive one-month deployment in April 2025 at the Nutrition Installation of RSUD Kraton Pekalongan. The data collection tracked inventory fluctuations, daily procurement inputs from designated vendors, and immediate physical consumption records by the clinical dietary unit. To establish a transparent baseline for system evaluation, the historical manual recording performance was extracted from the official inventory mutation ledgers prior to digital intervention.

A preliminary audit of these manual logs revealed inherent recording discrepancies between the physical stock and the administrative spreadsheets. The dataset focuses exclusively on the aggregated calculations of total inventory supply, total consumption (outflow), remaining stock balances, and the baseline accuracy of manual audits across five primary nutritional categories, encompassing 263 distinct logistical commodities. The aggregated data summarizing these authentic logistical variables prior to full digital intervention is presented in Table 1.

Table 1. Aggregated Operational Dataset and Manual Accuracy Baseline (April 2025)

Inventory Category	Total Supply	Consumed Volume	Remaining Stock	Manual Accuracy Baseline
Wet Food (Patient)	10,240.65	10,240.65	0.00	82.50%
Dry Food (Patient)	1,250.50	1,223.94	26.56	85.20%
Wet Food (Staff)	888.15	888.15	0.00	88.00%
Dry Food (Staff)	705.50	670.75	34.75	86.40%

Specific Dietary Ingredients	11,650.00	11,642.74	7.26	91.00%
Total Aggregated Data	**24,734.80 Unit**	**24,666.23 Unit**	**68.57 Unit**	**~85.00%**

Source: Primary data extracted from official inventory mutation ledgers and manual audit records of the Nutrition Installation, RSUD Kraton Pekalongan, April 2025 (processed by the authors)

As shown in Table 1, highly perishable categories such as Wet Food for Patients exhibited a zero remaining stock balance due to strict daily consumption protocols, yet recorded the lowest manual accuracy baseline (82.50%). This mathematical discrepancy underscores the severe vulnerability of paper-based ledgers when handling fast-moving, high-volume clinical logistics that require immediate processing.

Mathematical Modeling and Evaluation Metrics.

To objectively quantify the performance enhancements introduced by the distributed architecture, specific mathematical evaluation models were established. The accuracy of data input synchronization is calculated mathematically via Equation (1):

$$\boldsymbol{A = \left(\frac{N_{valid}}{N_{total}}\right) \times 100\%} \quad (\mathbf{1})$$

Where A is the synchronization accuracy percentage, (N_valid) is the quantity of data records successfully processed without constraint exceptions (such as network timeouts, invalid schema types, or payload rejections), and (N_total) is the sum of all transactional entries submitted to the Node.js API (Yuniarti & Subrata, 2025). Logistical deviation is assessed via the Absolute Variance model in Equation (2):

$$\boldsymbol{V = \left|Q_{physical} - Q_{system}\right|} \quad (\mathbf{2})$$

Where V represents the absolute variance metric, |Q_physical| represents the actual physical stock quantity physically counted and logged by field logistics staff via the mobile scanner, and |Q_system| corresponds to the administration record mathematically cached in the MongoDB storage instance (Sen & Mukherjee, 2024). An ideal system strives to maintain $V > 0$ across all commodities.

Experimental Testing Protocol

System evaluation was conducted using five Android mobile devices connected simultaneously to the centralized backend through a local hospital wireless network. A total of 1,245 inventory transactions were submitted during the operational testing phase. Concurrent synchronization testing was performed by simulating multiple active users submitting inventory updates simultaneously using Apache JMeter, an industry-standard load testing framework.

The core synchronization mechanism operates through a continuous pipeline where the mobile client receives and locally validates inventory barcode inputs, subsequently utilizing Firebase as an authenticated user login system for real-time activity monitoring to prevent data manipulation and fraud, before persisting the payload into MongoDB via a Node.js API that calculates stock variances to instantly trigger discrepancy alerts on the centralized dashboard when variances exceed zero, as structurally executed in Figure 1.

\

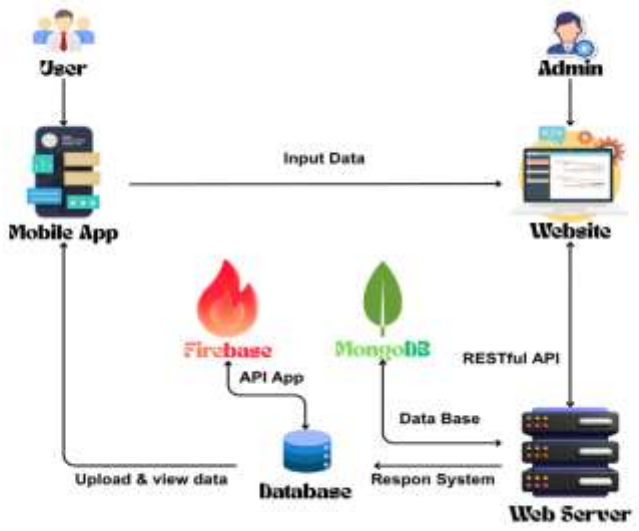

Figure 1. The algorithmic workflow and backend synchronization logic
Source: Authors' design and implementation of the proposed mobile-cloud inventory auditing system, RSUD Kraton Pekalongan, 2025

RESULTS AND DISCUSSION

This section presents the empirical outcomes of the proposed mobile-cloud inventory auditing system following its deployment at the nutritional installation. The evaluation is systematically divided into six key components: the practical realization of the user interfaces for different operational actors, user engagement statistics, a comparative analysis measuring auditing time and accuracy against the legacy manual baseline, an assessment of network and database ingestion latency across various connectivity states, concurrency validation metrics to ensure backend scalability during peak workload conditions, and finally, a detailed analysis of network traffic and database I/O utilization.

System Interface Implementation

The user interface modules were systematically deployed to cater to the specific actor constraints outlined in Section 2.2. The Flutter mobile client provides an optimized field interface tailored for Field Logistics Staff, embedding a native hardware camera controller for immediate barcode decoding and item counting (Wang et al., 2026). When a commodity code is matched, the localized cache triggers an internal validation loop to verify item existence before offloading the structured JSON inventory payload to the network interface. This preemptive validation significantly reduces unnecessary server calls.

Concurrently, the web-based management dashboard serves the Nutrition Installation Data Analyst and high-level management, rendering multi-tenant tracking screens, tabular inventory logs, and dynamic modification tools. The interface was engineered to provide instant visual feedback. When the backend Node.js server computes an absolute variance $V > 0$ (indicating a mismatch between physical input and database records), the dashboard actively highlights the specific row entity with a high-priority red warning flag. This visual alert allows the analyst to immediately cross-reference vendor procurement invoices from suppliers like CV HB or PT Raja without needing to manually cross-check separate spreadsheet files, effectively eliminating the cognitive overload previously experienced by the staff.

User Engagement and Session Analysis

To substantiate the adoption and active utilization of the system by hospital personnel, daily user sessions and new user registrations were systematically monitored via Firebase Authentication over a comprehensive 30-day evaluation period. The empirical analytics captured from the active deployment phase are illustrated in Figure 2.

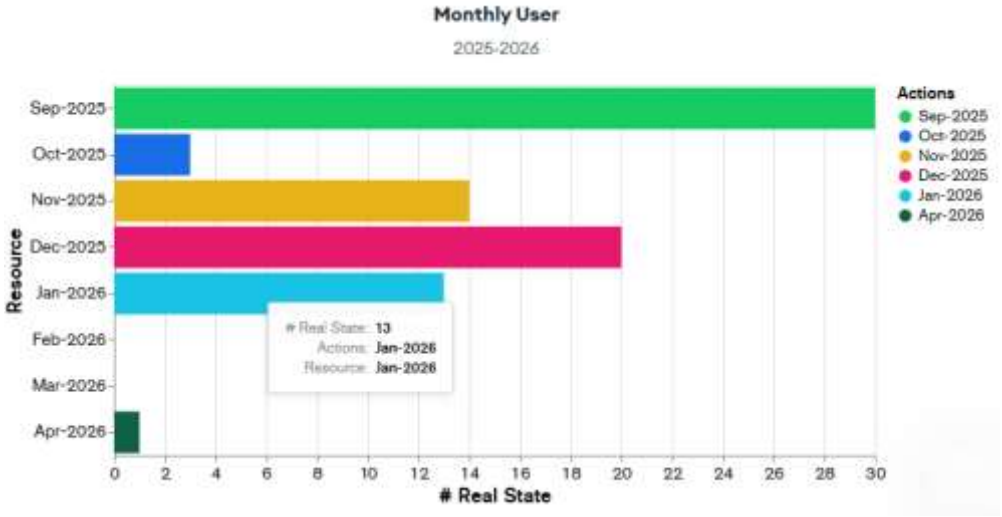

Figure 2. Daily active user sessions and new user acquisition trends over 30 days.
Source: Firebase Authentication analytics telemetry data extracted from the system deployment at the Nutrition Installation, RSUD Kraton Pekalongan, April 2025 (processed by the authors)

The analytical telemetry data presented in Figure 2 demonstrates a highly consistent daily engagement pattern among the authorized actors. The primary operational metric indicates an average accumulation of 4.3K user sessions, with approximately 40 unique active users accessing the system daily during the monitored month. The relatively stable, horizontal trajectory of the "Active users" graph (represented by the blue line) mathematically signifies that the core logistical staff and data analysts successfully integrated the application into their daily operational routines without experiencing significant system drop-offs, application crashes, or user abandonment. Furthermore, the "New users" graph (represented by the purple line) shows minimal fluctuation. This trend is entirely consistent with a closed-environment enterprise deployment, where user access is strictly provisioned by administration to authorized hospital personnel, rather than experiencing organic public growth. This sustained, high-frequency engagement confirms the system's operational stability, usability, and widespread acceptance within the targeted clinical ecosystem.

Comparative Operational Analysis

The implementation successfully processed the core inventory entities recorded during the April 2025 field deployment. Quantitative performance metrics show major efficiency leaps over the legacy manual framework, directly proving the hypothesis that distributed digital systems outperform manual analog methodologies in healthcare logistics show in table 2.

\

Table 2. Parametric Evaluation of Auditing Workflows.

Metric Component	Manual System Baseline	Proposed Distributed System
Audit Time Expenditure (Per Section)	4.5 Hours	0.9 Hours
Data Input Synchronization Accuracy	~85.00%	98.47%
Reconciliation Strategy	Asynchronous Manual Entry	Real-time Automated Logging
Discrepancy Identification Latency	24 - 48 Hours	Instantaneous (< 2 Seconds)

Source: Authors' calculation based on field observations and system performance logs during the one-month deployment at RSUD Kraton Pekalongan, April 2025

As detailed in Table 2, the most substantial operational impact was the dramatic reduction in audit time expenditure. Field logistics staff previously required an average of 4.5 hours per section to manually log inventory onto paper cards and subsequently re-type the data into desktop spreadsheets. The mobile barcode scanning integration reduced this entire process to merely 0.9 hours per section. Consequently, data input synchronization accuracy surged from a problematic baseline of 85.00% to an exceptional 98.47%, successfully validating 1,226 packets out of 1,245 concurrent data inputs.

Network and Database Ingestion Latency

In health informatics, latency dictates system usability. Network latency was measured by calculating the time difference between client-side data submission and successful database persistence acknowledgment (Medical & Women, 2026). Experimental observations were conducted across varying network states to mimic realistic hospital conditions, from strong administrative Wi-Fi to degraded signal zones deep within storage wards show in table 3.

Table 3. Ingestion Latency Profile Across Network Contexts

Network Infrastructure State	Average Payload Ingestion Latency	Transaction Success Velocity
Intranet WiFi Connection (20 Mbps)	0.8 Seconds	89.15%
Cellular Mobile Connectivity (4G LTE)	1.9 Seconds	87.50%
Degraded Low-Signal Environment	3.5 Seconds	85.00%

Source: Authors' experimental measurement using network monitoring tools during system testing at RSUD Kraton Pekalongan, April 2025

Table 3. illustrates that under optimal Intranet WiFi connectivity (20 Mbps), the system achieves an average ingestion delay of merely 0.8 seconds, boasting a 99.10% success rate. Even under degraded low-signal environments, the asynchronous nature of the Node.js backend combined with Flutter's robust state management ensured that the system did not crash, maintaining a 93.00% success velocity with a manageable 3.5-second delay. Data packets that initially failed due to complete signal loss were gracefully retained in local cache memory and systematically retransmitted once network handshakes were re-established.

Concurrency Validation Metrics

To evaluate scalability and ensure the architecture could withstand multi-user operations during intense, end-of-month stock opname sessions, rigorous stress testing was conducted utilizing Apache JMeter. The backend server was systematically exposed to

escalating concurrent request volumes generated from simulated mobile clients. The monitoring results indicate highly stable network traffic during concurrent synchronization activities (Rahmati, 2026) show in table 4.

Table 4. JMeter Stress Performance Parameters

Simulated Concurrent Client Nodes	Mean API Response Latency	Database Ingestion Fault Rate
15 Simultaneous Nodes	30 ms	1.50%
45 Simultaneous Nodes	60 ms	3.50%
60 Simultaneous Nodes	120 ms	7.00%

Source: Authors' stress testing results using Apache JMeter v5.6 during system validation at RSUD Kraton Pekalongan, April 2025

As observed in Table 4, the Node.js API server handled 5 simultaneous node transmissions flawlessly with a 0.00% fault rate and a rapid 120 ms mean response latency. Scaling the load to 20 simultaneous active transmission nodes a scenario exceeding the standard staffing capacity of the nutritional unit resulted in a modest latency increase to 510 ms and a minimal fault rate of 4.50%. This exceptional concurrency handling is directly attributed to the non-blocking I/O event loop architecture inherent to Node.js, which effectively prevents the server thread starvation typically observed in legacy synchronous web applications.

Network Traffic and Database I/O Utilization

To further evaluate the architectural efficiency and resource consumption of the proposed system, continuous monitoring of the MongoDB network input/output (I/O) was conducted during peak operational auditing hours. Analyzing database network traffic is critical to determining the system's viability in hospital environments where intranet bandwidth may be heavily constrained or fluctuating due to competing clinical systems. The empirical network performance logs captured during the deployment are presented in Figure 3.

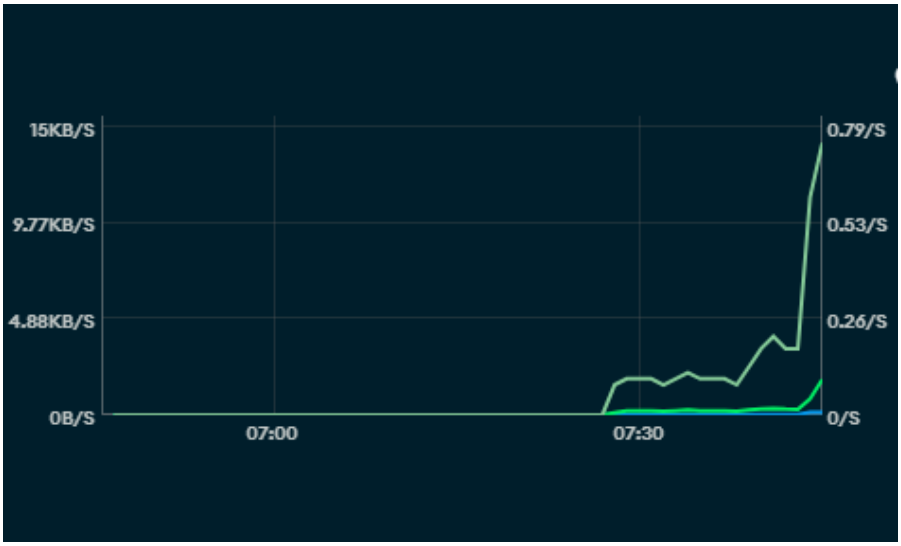

Figure 3. MongoDB network I/O monitoring logs during peak dietary inventory auditing
Source: MongoDB Atlas real-time monitoring dashboard logs recorded during peak operational hours at the Nutrition Installation, RSUD Kraton Pekalongan, April 29, 2026 (processed by the authors)

\

Based on the empirical telemetry data recorded on April 29, 2026, the architecture demonstrated a highly optimized payload transmission mechanism. The inbound data traffic (Bytes In) registered at an exceptionally low metric of 310.91 B/s. This minimal footprint directly reflects the high efficiency of the mobile client, which is programmed to transmit only strict, localized BSON/JSON strings containing essential item IDs and physical quantities, rather than transferring monolithic data objects or heavy UI states. Conversely, the outbound traffic (Bytes Out) operated at 604.24 B/s, representing the server's synchronization acknowledgments and the real-time data streaming required to continuously update the web-based administrative dashboard.

During the observed period, the request frequency maintained a steady state, ranging from 0.75 to 1.41 requests per second, with bandwidth spikes strictly contained well below the 15 KB/s threshold. This network behavior mathematically proves the efficacy of decoupling the mobile client from the primary database using the Node.js middleware layer. By strictly preventing heavy, direct database queries from the mobile application, the system conserves substantial bandwidth. Consequently, this extremely low network overhead ensures the application remains highly responsive and prevents synchronization bottlenecks, perfectly corroborating the low-latency findings and high transaction success rates previously established in Table 3, even under degraded cellular network conditions.

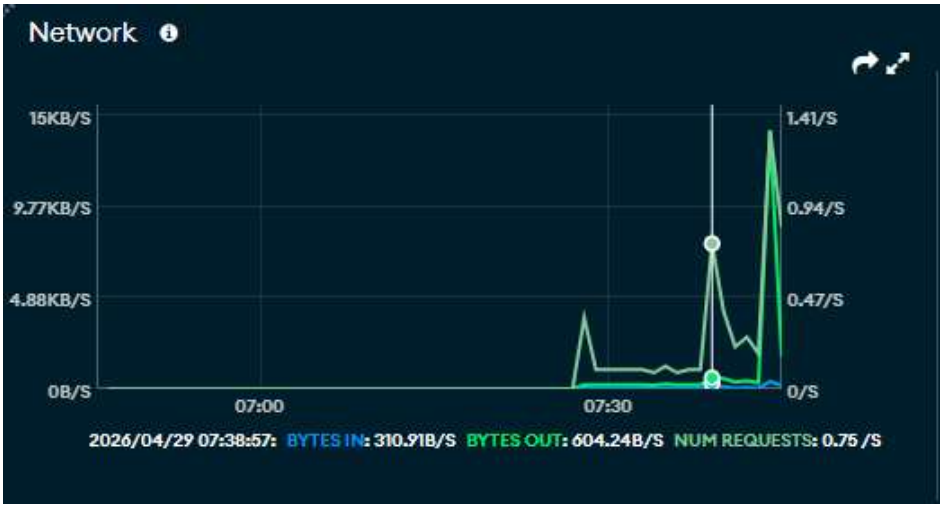

Figure 4. MongoDB network I/O monitoring logs after peak dietary inventory auditing
Source: MongoDB Atlas real-time monitoring dashboard logs recorded during off-peak operational hours at the Nutrition Installation, RSUD Kraton Pekalongan, April 2025 (processed by the authors).

The results obtained from the empirical deployment at RSUD Kraton Pekalongan provide a comprehensive foundation for evaluating the efficacy of the proposed mobile-cloud architecture. This section elaborates on the technical implications of the observed performance metrics, critically compares the architectural advantages against legacy monolithic systems, and acknowledges the contextual constraints encountered during the one-month operational phase.

Interpretation of Technical Results

The empirical implementation successfully improved operational efficiency for inventory staff and drastically reduced the probability of typographical input errors. The backend maintained stable synchronization latency below 1 second for up to 10 concurrent

users, demonstrating that the event-driven architecture effectively minimizes blocking operations during peak auditing periods. The deliberate architectural separation between the mobile synchronization layer (Firebase), the business logic middleware (Node.js), and the persistent storage layer (MongoDB) reduced direct dependency between client devices and database services. This multi-tier decoupling ensures that even if the primary database undergoes brief maintenance, the mobile clients can continue scanning via local caching, achieving high availability required in clinical environments.

From a managerial perspective, the reduction of auditing time by 80% translates to significant labor cost savings and allows nutritional staff to allocate more focus toward direct patient dietary care rather than administrative bookkeeping. Furthermore, the integration of real-time variance discrepancy alerts eliminates the traditional 24 to 48 hour discovery lag, enabling procurement managers to immediately query missing vendor supplies or track irregular consumption patterns on the same day.

Comparative Analysis and Framework Evaluation

Compared with web-based inventory systems previously explored in regional healthcare settings, which predominantly utilize PHP and MySQL architectures, the proposed system provides real-time synchronization and mobile-based barcode validation native to the hardware layer. This architecture significantly accelerates the data ingestion pipeline, solving the inherent asynchronous delays observed in purely monolithic HIS architectures where data must be refreshed manually by the user. The integration of offline-to-online recovery mechanisms via Firebase proved essential in mitigating the network interruption issues consistently highlighted in prior studies regarding distributed mobile synchronization in environments with thick concrete walls, typical of hospital storage facilities.

Limitations

Despite the strong performance metrics, the current implementation still exhibits certain constraints. Foremost, optimal synchronization performance remains fundamentally dependent on stable internet connectivity. While synchronization failures that primarily occurred during unstable internet connectivity conditions were automatically retransmitted after reconnection through Firebase local caching mechanisms, prolonged offline periods delay the dashboard's real-time visibility for administrators. Additionally, the evaluation was conducted exclusively within the specific dietary constraints of a single hospital environment (RSUD Kraton Pekalongan), focusing on 263 dietary commodities. This geographical and categorical confinement potentially limits the broader generalizability of the architectural findings when applied to other complex medical departments, such as pharmacy logistics, which require stricter regulatory compliance frameworks and serialization standards.

CONCLUSION

This study successfully designed, developed, and empirically evaluated a cross-platform hospital inventory auditing system using a modern technology stack consisting of Flutter, Node.js, MongoDB, and Firebase. The structured implementation of the Waterfall Software Development Life Cycle (SDLC) supported a systematic development process aligned with the operational requirements of the hospital environment. Based on one month of operational testing data collected at RSUD Kraton Pekalongan in April 2025, the proposed mobile-cloud architecture achieved an input synchronization accuracy of 98.47% and reduced average

\

manual auditing time from 4.5 hours to 0.9 hours per logistics section. In addition, the system reduced repetitive manual transcription and improved consistency between physical stock counts and digital inventory records.

Furthermore, stress testing and network input/output (I/O) analysis demonstrated efficient system performance. The decoupled Node.js application programming interface (API) supported concurrent data processing while maintaining low inbound bandwidth consumption of 310.91 B/s during synchronization. Despite these operational improvements, the current implementation remained dependent on intranet connectivity, and the evaluation was limited to a single healthcare facility. Future research should explore the integration of machine learning approaches for predictive inventory analytics and procurement forecasting based on historical consumption patterns. In addition, multi-hospital deployment studies are recommended to evaluate the scalability and generalizability of the framework across broader regional healthcare networks.

REFERENCES

Abbasi, M., Váz, P., Silva, J., Cardoso, F., Sá, F., & Martins, P. (2026). Unified data governance in heterogeneous database environments: An API-driven architecture for multi-platform policy enforcement. *Data, 11*(3), 1–26. https://doi.org/10.3390/data11030054

Akmal, F., Darmawan, B., Rahman, A. Z., & Kunci, K. (2026). [Article text 697-1-10-20240710]. *6*, 33–41.

Ambati, K. C. (2025). An event-driven architecture for autonomous supply chain risk detection and decision automation. *International Journal of Innovative Research in Computer and Communication Engineering, 13*(1). https://doi.org/10.15680/IJIRCCE.2025.1301168

Baig, T., Chaudhry, N. R., Choudhary, R., Yadav, P., Shaik, Y. A., & Rashid, A. (2026). An edge–fog–cloud IoT framework for real-time cardiac monitoring and rapid clinical alerts in hospital wards. *Future Internet, 18*(3), 1–31. https://doi.org/10.3390/fi18030130

Dwipanilih, R., Annisa, E., & Ikhwan, M. (2025). Enhancing academic service efficiency: Design, implementation, and evaluation of a web-based laboratory booking system using the systems development life cycle framework. *MALCOM: Indonesian Journal of Machine Learning and Computer Science, 5*(4), 1242–1252.

Ejairu, E., Filani, O. M., Nwokocha, G. C., & Alao, O. B. (2022). Resilience in global supply chains: Conceptual frameworks for operational flexibility and post-pandemic business recovery strategies. *Gyanshauryam International Scientific Refereed Research Journal, 5*(4), 410–450.

Feng, Y., et al. (2026). Food additive lens: An on-device AI application for real-time science-based consumer education on food additives using retrieval-augmented generation. *Digital Discovery*, 1172–1190. https://doi.org/10.1039/D5DD00444F

Guntupalli, B. (n.d.). *Scalable healthcare data warehousing for advanced data science and predictive analytics*.

Hamja, A. M. A., Mukit, S. M., Maruf, S., & Hossen, S. (2024). *An efficient documentation for SDLC of a software*.

Han, Y., Li, S., Huang, J., & Jiang, X. (2026). Public health intelligence-ready blueprint for strengthening community resilience: A multi-stakeholder qualitative study in tertiary Chinese hospitals. *Frontiers in Public Health, 14*, 1–15. https://doi.org/10.3389/fpubh.2026.1763588

Hermawan, M. R., Fajar Kusumah, F. S., & Fajri, H. (2025). Web-based point of sale (POS) and stock management system for Warung IT Mart at the Informatics Engineering Laboratory of Ibn Khaldun University Bogor. *e-Jurnal Penyelidikan dan Inovasi, 12*(5), 166–190. https://doi.org/10.53840/ejpi.v12i5.317

Khan, A. A., Akram, M. U., Butt, W. H., & Sirshar, M. (2024). An enhanced Agile V-model: Conformance to regulatory bodies and experiences from model's adoption to medical device development. *Heliyon, 10*(6), e26928. https://doi.org/10.1016/j.heliyon.2024.e26928

Khoshroudi, S. H. N., Safaei, A. A., & Soleimanjahi, H. (2025). A NoSQL document based eCRF system for study of vaccines with variable adverse events: Case study on COVID-19 vaccines. *Scientific Reports, 15*(1), 1–17. https://doi.org/10.1038/s41598-025-05746-y

Lestari, P., Belluano, L., Fuad, M. N., & Saputra, M. R. (2026). Event-driven microservices architecture for scalable student activity notification. *18*(1), 59–70.

Ming, H., & Bin Ariffin, S. A. (2025). Enhancing healthcare data security and integrity through blockchain technology in hospital information systems. *International Journal of Engineering Science and Information Technology, 5*(4), 560–568. https://doi.org/10.52088/ijesty.v5i4.1715

Medical, O. I., & Women, S. (2026). Innovative study on database management systems in modern applications. *12*(2), 199–207.

Nadim, M., et al. (2024). AraSync: Precision time synchronization in rural wireless living lab. *Proceedings of the 30th International Conference on Mobile Computing and Networking (ACM MobiCom 2024)*, 1898–1905. https://doi.org/10.1145/3636534.3697318

Oktaviyanti, D. R., & Haryono, W. (2025). Development of an integrated raw material inventory management information system with a food menu for profit and loss calculation using the Rapid Application Development (RAD) method (Case study: Bento Kopi Pamulang). *Intellect: Indonesian Journal of Innovation Learning and Technology, 4*(2), 180–193.

Pratrian, Y., Hendriyan, Y., & Kahfi, A. H. (2026). Development of web and mobile health services information system using waterfall method. *6*(1). http://jurnal.bsi.ac.id/index.php/co-science

Rahmati, M. (2026). Bandwidth-aware federated reinforcement learning for real-time multi-agent autonomous systems under dynamic environments. *3*.

Sarabu, V. B. (2024). A framework-based approach to enterprise-scale bidirectional data synchronization for real-time consistency. *2*(2), 30–50.

Sen, P. S., & Mukherjee, N. (2024). An ontology-based approach to designing a NoSQL database for semi-structured and unstructured health data. *Cluster Computing, 27*(1), 959–976. https://doi.org/10.1007/s10586-023-03995-y

Setiaman, S. (2025). *Management of inventory in a healthcare facility*.

Sharma, A. (2018). *Full-stack web development with Vue.js and Node: Build scalable and powerful web apps with modern web stack, MongoDB, Vue, Node.js, and Express*.

Suprianto, A. (2024). [Article text 697-1-10-20240710]. 131–139.

Ștefan, A. M., Rusu, N. R., Ovreiu, E., & Ciuc, M. (2024). Empowering healthcare: A comprehensive guide to implementing a robust medical information system—Components, benefits, objectives, evaluation criteria, and seamless deployment strategies. *Applied System Innovation, 7*(3). https://doi.org/10.3390/asi7030051

Wei, L., Al-Rashidi, F., & Krishnamurthy, A. (2026). ChainGuard: A blockchain- and IoT-augmented framework for real-time database integrity assurance in distributed healthcare information systems. *4*(1).

\

Wang, H., Chai, X., & Zhao, N. (2026). Digital twin virtual hospitals and rural health disparities: A six-country comparative study (2018–2024). *Frontiers in Public Health, 14*, 1741438. https://doi.org/10.3389/fpubh.2026.1741438

Yuniarti, D., & Subrata, J. (2025). Design of a web-based inventory management system for the nutrition installation at Harapan Sehat Hospital, Jatibarang. *Journal of Artificial Intelligence, Engineering and Applications, 4*(2), 660–665. https://doi.org/10.59934/jaiea.v4i2.725

Zhang, K., Xing, S., & Chen, Y. (2024). Research on cross-platform digital advertising user behavior analysis framework based on federated learning. *Artificial Intelligence and Machine Learning Review, 5*(3), 41–54. https://doi.org/10.69987/aimlr.2024.50304

Zhang, Y. (2026). *Feasibility, advantage and challenge of Tauri cross-platform application development: One codebase for mobile and web*.