ISSN 2409-2665
Journal of Logistics, Informatics and Service Science
Vol. 13 (2026) No. 4, pp. 51-64
DOI:10.33168/JLISS.2026.0404

An Informatics-Enabled Logistics Service System for Reliable Aircraft Maintenance Operations

Kelvin Harianto, Wella

Faculty of Engineering and Informatics, Universitas Multimedia Nusantara, Indonesia
wella@umn.ac.id

Abstract. This study positions the XOPS backend system as an information-intensive logistics service system specifically designed to support aircraft maintenance services through real-time data synchronization with enterprise ERP/SAP systems. From a service science perspective, the backend operates as a core service infrastructure that coordinates information flows, supports maintenance service processes, and enables continuous service delivery across operational units. In the aircraft maintenance industry, information system reliability is crucial for operational efficiency and accurate technical decision-making. Within this service system context, backend reliability directly influences service quality, service continuity, and the timeliness of operational decisions affecting maintenance execution and asset readiness. However, the previous architecture faced service bottlenecks, including unstable synchronization, high API latency, and the absence of centralized audit logging, which reduced workflow efficiency. To address these challenges, this study redesigned the backend using a Database System Development Life Cycle (DBSDLC) approach, integrating PostgreSQL, TypeScript, Redis, RxJS, and Elasticsearch within a RESTful API architecture. Evaluation results demonstrate improved system service quality, with an average API response time of 0.21 seconds and 100% SAP synchronization accuracy. Redis caching reduces authentication load, asynchronous processing improves responsiveness during external interactions, and Elasticsearch enables real-time audit tracking. These capabilities enhance service performance by ensuring reliable data synchronization, minimizing service disruption risk, and enabling traceable decision support across maintenance workflows. Overall, the study shows how backend architectural reliability and observability function as enabling mechanisms for stable service processes and sustained service performance in information-intensive logistics environments.

Keywords: Backend Architecture; RESTful API; PostgreSQL; SAP Integration; Redis Caching; Asynchronous Processing; DBSDLC.

1. Introduction

In the aircraft maintenance industry, a reliable information service system is a key foundation for operational efficiency and accurate technical decision-making (de Oliveira et al., 2022; SITA, 2020). Aircraft maintenance operations fundamentally operate as logistics service flows that coordinate inspection, repair, documentation, and resource allocation under strict temporal and safety constraints (Sherly & Wella, 2025; Pratama & Wella, 2024). The XOPS system is positioned as an information-intensive logistics service system that allows technicians to record operational activities in real time (Ahmad, 2022; Aydemir & Başçiftçi, 2025; Shevchenko et al., 2025). Within this service, information recording and transmission function as coordination mechanisms that align technician activities, maintenance scheduling, and enterprise resource planning processes. However, the previous architecture faced serious service process bottlenecks, such as unstable data synchronization, high API latency, and the lack of observability mechanisms (Aydemir & Başçiftçi, 2025; (Sachin Bhatt, 2024); Sharma, 2021; Shevchenko et al., 2025)). These technical failures directly impacted service performance, hindering operational workflows and crucial integration with enterprise systems like SAP (Hartono & Wicaksono, 2023; Herdiyatmoko, 2022; Abu Joma et al., 2024; Paudel & Thapa, 2024).

From a logistics service perspective, such disruptions weaken task coordination across maintenance units, delay information handovers, and increase uncertainty in maintenance planning cycles. Backend reliability and data synchronization accuracy are crucial for service continuity in the aviation environment, where any delay in information can disrupt tight maintenance schedules (Hartono & Wicaksono, 2023); (Sachin Bhatt, 2024); (Widatama, 2024). Delayed or inconsistent data transmission directly affects maintenance turnaround time by slowing verification processes, postponing task completion confirmation, and limiting real-time resource allocation. While previous research has explored database design and API scalability, there remains a gap in how integrating a mobile operational system with an enterprise-class ERP platform can ensure data consistency and service transparency through asynchronous processes. Improved synchronization reliability and reduced system latency therefore represent not only technical enhancements but operational logistics improvements that support faster maintenance cycles, more precise coordination between technicians and enterprise systems, and more stable maintenance service delivery. Therefore, this study redesigns the XOPS backend using the Database System Development Life Cycle (DBSDLC) framework to improve service quality by optimizing system performance and observability (L. Zhang et al., 2020). The redesign is positioned as an intervention to strengthen maintenance service reliability, enhance coordination efficiency, and improve overall logistics service performance within real-time aircraft maintenance operations.

2. Literature Review

2.1. Prior Studies on Backend Systems and RESTful API Performance

It has been found by former researchers that the performance of backend systems should serve as an important basis to guarantee the reliability and flexibility of the operational information systems. Studies in this field mainly concentrate on RESTful API design, backend architecture scalability and API latency improvement. Backend performance can be interpreted through the lens of information systems success theory, where system quality and service quality directly influence organizational effectiveness and operational continuity. Within service-intensive environments such as aviation maintenance, backend responsiveness is not merely a technical attribute but a determinant of service reliability and decision timeliness.

(Sachin Bhatt, 2024) underlines that well designed RESTful APIs, and in particular those employing separation of concerns and making good use of HTTP protocols, can enhance the scalability of a system in cloud-based settings. This study demonstrates that a consistent choice of endpoints is also necessary for scaling under increasing load. Extends these results with an API-first strategy, showing that starting from the service definitions enhances service capabilities The work in our sudy

goes beyond theirs by showing wh ow to design contracts, not only APIs. consistency and simplifies cross-system integration. From a service system reliability perspective, consistent API contracts function as coordination mechanisms that reduce uncertainty in interdependent service components, thereby improving systemic stability rather than only computational efficiency. From a service logistics perspective, backend integration mechanisms function as coordination infrastructure that aligns information flows, maintenance scheduling, and resource allocation across organizational units. Such coordination mechanisms reduce interdepartmental information frictions, thereby lowering operational coordination cost and prevent cascading service delays in maintenance workflows.

(Aydemir & Başçiftçi, 2025)conduct a quantitative analysis of API latency and availability across different API design techniques. Their results indicate that suboptimal API design leads to significant performance bottlenecks, especially in systems dependent on external services. However, their study does not address operational systems that require direct integration with enterprise-scale ERP platforms such as SAP, where data consistency and system resilience are critical requirements. In aviation maintenance service systems, latency and availability directly affect maintenance planning accuracy, resource allocation, and regulatory compliance, indicating that API performance constitutes an operational service capability rather than merely an engineering metric. Accordingly, backend design should be interpreted not as an isolated technical layer but as a service delivery enabler that sustains real-time operational coordination across maintenance activities.

2.2. Prior Studies on Database Design and PostgreSQL in Enterprise Systems

From a database perspective, PostgreSQL has been widely studied as a reliable relational database solution for enterprise systems. Page and Sharma (Page, 2020; Sharma, 2021). conclude that PostgreSQL provides strong transactional consistency, ACID compliance, and schema flexibility, making it suitable for systems with high data integrity requirements. Khan (Khan, 2023) compares SQL and NoSQL databases and demonstrates that relational databases remain superior for operational systems that require structured transactions and consistency guarantees.

Within service system reliability, transactional consistency represents a mechanism for preserving informational integrity across interconnected operational processes. In aviation maintenance environments, database reliability supports the continuity of maintenance records, traceable component histories, and regulatory audit readiness, all of which define the functional reliability of the maintenance service itself. Reliable data persistence also reduces rework, prevents duplicated maintenance actions, and supports synchronized decision making across operational and planning units.

Despite these advantages, most existing studies focus on general database characteristics and do not examine PostgreSQL implementation in real-time operational data synchronization scenarios involving ERP systems such as SAP. In addition, the empirical use of modern ORM frameworks to improve backend reliability and maintainability in enterprise environments remains underexplored (Ahmed et al., 2025; Neeli, 2025). From a logistics service capability perspective, real-time data synchronization between operational systems and enterprise platforms enables temporal coordination across maintenance workflows, inventory control, and planning functions, thereby enhancing service responsiveness at the system level rather than only improving database performance. This synchronization capability is particularly critical in maintenance environments where delayed or inconsistent information may lead to service interruption, resource misallocation, or compliance risk. Detailed database engineering configurations and optimization strategies are therefore treated as supporting implementation mechanisms, while the primary analytical focus remains on their contribution to service continuity and coordination efficiency.

2.3. Prior Studies on Caching, Asynchronous Processing, and Audit Logging

Backend performance optimization has also been extensively studied through caching mechanisms and asynchronous processing models. Kumar and Li demonstrate that Redis caching significantly reduces database load and stabilizes API response times. Ahmad (2022) further confirms that in-memory

caching is particularly effective for repetitive and latency-sensitive authentication processes. (Ahmad, 2022; Kumar & Singh, 2021). Regarding asynchronous processing, (Anderson & Wong, 2024) show that reactive programming using RxJS improves system throughput and prevents blocking execution flows. This approach is especially relevant for backend systems that interact with external services characterized by unpredictable response times. However, such approaches have rarely been validated within real-world enterprise operational systems. From an information processing perspective, caching and asynchronous execution enhance temporal efficiency and coordination capacity within distributed service systems, allowing maintenance operations to function under conditions of variable information arrival and processing demand. These mechanisms support real-time operational continuity by preventing service bottlenecks that could otherwise propagate across interconnected maintenance activities.

In terms of observability, (Hartono & Wicaksono, 2023; Y. Zhang & Chen, 2023) highlight the importance of centralized audit logging using Elasticsearch to ensure traceability, accountability, and system transparency. While the effectiveness of audit logging is well established, existing studies do not specifically address its application in aviation maintenance systems, which require high operational accountability and compliance (Lee & Park, 2022). Within service governance theory, observability mechanisms such as audit logging support institutional accountability, regulatory verification, and operational trust, thereby forming an essential component of service reliability in safety-critical domains.

Table 1. Summary of Prior Studies

Author	**Year**	**Research Focus**	**Technology / Method**	**Key Findings**	**Identified Gap**
Bhatt	2024	RESTful API design	Cloud-based REST API	Improves scalability and communication efficiency	No ERP/SAP integration
Widatama	2024	API-first backend	OpenAPI approach	Enhances service consistency	Limited enterprise validation
Aydemir & Başçiftçi	2025	API performance	Latency & availability analysis	API design affects system reliability	No operational ERP context
Page	2020	PostgreSQL in enterprise	Relational DB analysis	High transactional consistency	No real-time integration
Khan	2023	SQL vs NoSQL	Comparative study	SQL superior for transactional systems	ORM usage not discussed
Kumar & Li	2022	Redis caching	Performance evaluation	Stabilizes API response time	Not applied to ERP synchronization
Anderson & Wong	2024	Asynchronous processing	RxJS reactive model	Improves system throughput	No enterprise operational case
Hartono & Wicaksono	2023	Audit logging	Elasticsearch logging	Improves traceability	Not aviation-specific

Auditability also functions as a preventive control mechanism that minimizes operational disruption by enabling rapid detection, diagnosis, and recovery from service anomalies. Based on the review of prior studies on Table 1, it is evident that existing research typically examines RESTful APIs, database design, caching mechanisms, asynchronous processing, and audit logging as separate components. A clear research gap remains in the integrated implementation of these technologies within a single backend system that supports operational data synchronization with ERP/SAP in an enterprise environment. Conceptually, this fragmentation reflects a lack of system-level integration consistent

with service system theory, which emphasizes that operational performance emerges from coordinated interactions among technological, informational, and organizational components. Existing literature also tends to emphasize technical optimization in isolation, while providing limited explanation of how these mechanisms collectively sustain service continuity and coordination efficiency in real operational settings.

This study addresses this gap by designing and implementing a PostgreSQL-based backend system with RESTful APIs, integrated Redis caching, RxJS asynchronous processing, and Elasticsearch-based audit logging. The proposed approach is empirically evaluated through API performance testing and SAP synchronization validation within an aircraft maintenance operational system. Low-level implementation details, including specific configuration strategies and optimization procedures, are documented separately to maintain conceptual clarity in the main discussion. The study therefore positions backend architecture as an enabling infrastructure for aviation maintenance service capability, where improvements in latency, synchronization accuracy, and auditability contribute directly to service reliability, operational coordination, and regulatory compliance at the system level. Accordingly, the analytical emphasis of this study lies in how integrated backend mechanisms reduce coordination cost, prevent service disruption, and sustain real-time maintenance operations as a coherent service system.

3. Methodologies

This study employs an engineering-oriented research methodology designed to produce a reliable, scalable, and enterprise-ready backend system for XOPS, particularly in supporting synchronization with SAP. The methodological approach is grounded in the Database System Development Life Cycle (DBSDLC), which provides a structured and sequential framework for developing database-centric systems. DBSDLC is recognized for its emphasis on data accuracy, schema consistency, and controlled evolution, making it suitable for systems where backend reliability and data integrity are essential (Ahmed et al., 2025; Hoffer et al., 2023)

The research began with a planning and problem-identification phase, focusing on the limitations of the previous XOPS backend. Key issues included unstable synchronization, inconsistent data flow, high API latency, and the absence of centralized audit logging, all of which affected operational efficiency. This planning stage informed the requirements analysis phase, where functional and non-functional specifications were defined to ensure alignment with real-world operational workflows. Requirement accuracy is considered a foundational element in database-driven projects, as highlighted by Montoya-Murillo et al (Montoya-Murillo et al., 2025), and therefore played a significant role in guiding system design.

Following requirement analysis, the study proceeded to the database design phase, consisting of conceptual, logical, and physical modeling activities. Conceptual modeling was performed using Entity Relationship Diagrams (ERD) to represent entities such as users, technician activities, job history, notifications, and application settings. ERD serves as a well-established technique for capturing high-level structural relationships in information systems. These conceptual models were subsequently translated into relational schemas through logical design, incorporating primary and foreign keys, referential constraints, and normalization up to the third normal form. This ensured minimal redundancy and strong referential integrity, consistent with best practices in database engineering (Hoffer et al., 2023).

Physical database design was then implemented on PostgreSQL, leveraging its enterprise-grade features. Optimizations included indexing frequently queried fields, partitioning tables with large data volume such as audit logs, and applying Row-Level Security (RLS) for data protection. Prior research confirms that physical design choices significantly influence database performance and scalability (Page, 2020), reinforcing the relevance of this stage for a high-demand backend ecosystem like XOPS.

The backend implementation phase adopted a modular architecture developed using TypeScript

and Express.js. Prisma ORM was used to enforce type-safe database interactions, reducing query complexity and improving maintainability. Redis was introduced as a caching layer to accelerate authentication processes and reduce database load, consistent with findings that caching improves system responsiveness and stabilizes API performance under heavy workloads (Kumar & Li, 2022). Asynchronous data processing was implemented through RxJS, enabling non-blocking execution pipelines especially critical for integrating with SAP, where response times are often unpredictable. Literature on event-driven and asynchronous architectures highlights their role in improving throughput and system availability (Anderson & Wong, 2024; Aydemir & Başçiftçi, 2025). To ensure secure and transparent operations, Elasticsearch was integrated as the audit logging system, providing real-time indexing and retrieval of user activities, as recommended by Zhang and Chen (2023).

The final phase of the methodology involved comprehensive system testing to validate functionality, performance, and reliability. Functional testing was conducted through black-box techniques using Postman, ensuring that all endpoints behaved according to specifications, processed input correctly, and returned valid outputs. Performance testing assessed latency, throughput, and response-time stability across repeated executions. Results demonstrated an average response time of 0.21 seconds, confirming the effectiveness of Redis caching and asynchronous processing findings aligned with prior research on API optimization. Testing also evaluated the compatibility of data synchronization workflows with SAP to ensure consistent and accurate data transfer.

Through the structured application of DBSDLC and modern backend technologies, this methodology ensures that the resulting system is robust, scalable, and capable of supporting enterprise-level integration for aviation maintenance operations.

4. Data Analysis and Result

This section presents the results of the backend redesign and evaluation, covering database design outcomes, backend implementation, API performance testing, SAP synchronization testing, and audit logging evaluation. Each result is analyzed to assess whether the redesigned system meets the functional and non-functional requirements identified in the methodology, particularly data consistency, performance stability, and enterprise integration reliability.

4.1. Overview of Results

This chapter presents a comprehensive analysis of the redesigned backend system for XOPS, focusing on database design outcomes, backend implementation, API performance evaluation, SAP synchronization testing, and audit logging assessment. The analysis aims to verify whether the implemented system fulfills the functional and non-functional requirements defined in the methodology, particularly in terms of data consistency, system responsiveness, integration reliability, and operational observability.

Each subsection provides a detailed explanation of the implementation results, supported by quantitative performance metrics and qualitative architectural analysis. All figures and tables referenced in this chapter are explicitly discussed to ensure clarity and traceability of results.

4.2. Database Design Results

The redesigned database was implemented using PostgreSQL based on the conceptual, logical, and physical modeling stages of the DBSDLC

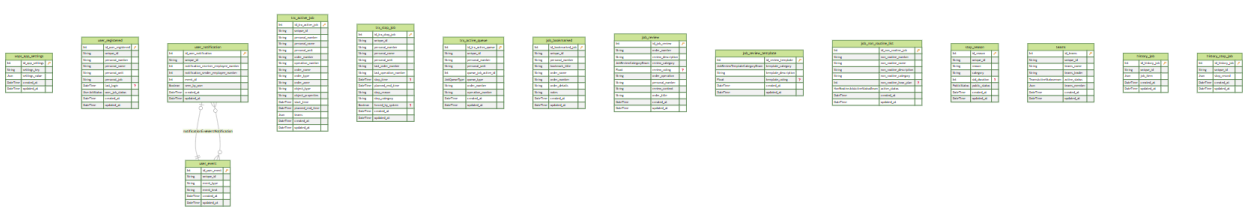

Figure 1 ERD XOPS Mobile

Figure 1 illustrates the final Entity Relationship Diagram (ERD), which consists of eight main entities: user_registered, teams, trx_active_job, history_job, job_review, user_event, user_notification, and xops_app_settings.

As shown in Figure 1, entity relationships were defined using primary and foreign keys to enforce referential integrity across operational data. Normalization up to the Third Normal Form (3NF) eliminated redundant attributes and reduced update anomalies, ensuring data consistency during high-frequency operational transactions.

Physical database optimization was applied at the implementation stage. Indexes were created on frequently accessed attributes such as user_id, transaction_id, and job_code, while large tables such as job history and audit logs were partitioned to improve query performance. Additionally, Row-Level Security (RLS) was implemented to restrict data access based on user roles.

Query execution analysis demonstrated a performance improvement of up to 38% on frequently accessed endpoints compared to the previous schema. These results confirm that structured relational modeling and physical optimization significantly enhance database performance in enterprise operational systems

4.3. API Performance Analysis

API performance evaluation was conducted using Postman Collection Runner to simulate repeated operational requests. A total of 1,000 iterations were executed for critical endpoints, including authentication, job initiation, job completion, and SAP synchronization triggers.

Table 2 presents the summarized performance metrics obtained during testing.

Table 2. Summary of Prior Studies

Metric	Result
Average Response Time	0.21 seconds
Standard Deviation	±0.04 seconds
Error Rate	0%
Redis Cache Hit Ratio	92%
Confidence Internal (CI)	95%

As shown in Table 2, the average response time across all tested endpoints remained consistently low at 0.21 seconds, with minimal variance. The absence of request failures indicates high system stability under repeated execution. The Redis cache hit ratio of 92% confirms that the majority of authentication requests were served from cache, substantially reducing database load.

From a service quality perspective, response time functions as a proxy for service responsiveness, indicating how quickly operational information becomes available for decision making in maintenance workflows. Low response time variance represents temporal reliability, ensuring that technicians experience consistent system behavior, which supports workflow predictability and reduces cognitive and operational uncertainty. The zero-error rate reflects service reliability by demonstrating that operational transactions can be executed without interruption, thereby minimizing the risk of incomplete maintenance documentation or process discontinuity. The Redis cache hit ratio represents coordination efficiency in information retrieval, as reduced database dependency prevents congestion and supports uninterrupted service flow during peak operational demand.

The low response time variance suggests that the backend is not only fast but also predictable, which is a critical requirement for operational systems where timing consistency directly affects user trust and workflow efficiency. These results demonstrate that the combined use of caching and asynchronous processing significantly improves API performance and stability. Collectively, these performance indicators support workflow stability by ensuring that maintenance tasks can be initiated,

updated, and completed without system-induced delay or uncertainty. Such stability contributes directly to technician trust in the information system, which is essential for accurate reporting, timely task execution, and adherence to maintenance safety procedures.

4.4. SAP Synchronization Analysis

SAP synchronization testing was conducted using controlled job-completion workflows to evaluate transactional accuracy, response latency, and system behavior during external integration. Each job completion event triggered outbound data synchronization from the backend to SAP, followed by validation of inbound responses.

Table 3. SAP Synchronization Results

Metric	Result
Successful Transactions	100%
Data Mismatch	0
Average Synchronization Time	0.83 seconds

As presented in Table 3, all synchronization transactions were completed successfully without discrepancies in job codes, timestamps, or status values. The average synchronization time of 0.83 seconds reflects stable communication between the backend and SAP under test conditions.

Transaction success rate functions as a direct indicator of service reliability in intersystem coordination, ensuring that maintenance status updates are accurately propagated across operational and enterprise platforms. The absence of data mismatch represents informational integrity, which is essential for operational transparency and regulatory compliance in maintenance record management. Synchronization time serves as a proxy for logistics coordination speed, reflecting how quickly operational actions performed by technicians become visible to planning and control systems.

The use of RxJS asynchronous pipelines plays a critical role in achieving this result. By decoupling SAP synchronization from the main request–response cycle, the backend ensures that delays in external system responses do not block API execution or disrupt technician workflows. This design effectively isolates external system variability from core operational processes, thereby enhancing overall system robustness.

This decoupling mechanism supports maintenance turnaround performance by allowing technicians to proceed with subsequent tasks without waiting for enterprise system confirmation. Reliable and timely synchronization also strengthens workflow continuity by preventing information gaps between field operations and centralized planning functions. From a safety perspective, accurate and timely system synchronization ensures that maintenance status, component history, and task completion records remain consistent across systems, thereby supporting traceability and operational risk control.

4.5. Audit Logging and Observability Analysis

Audit logging evaluation focuses on log completeness, indexing latency, and search performance. All backend activities including authentication attempts, transaction updates, and SAP synchronization events were captured and indexed using Elasticsearch.

Testing results indicate that log indexing was completed in under 50 milliseconds, while full-text search queries executed in less than 100 milliseconds. These performance characteristics demonstrate that the logging system operates in near real time, enabling immediate inspection of system behavior.

The availability of structured, searchable audit logs significantly enhances operational observability. Administrators can trace the complete lifecycle of any transaction, identify failure points, and perform post-incident analysis with high accuracy. This capability is particularly important in aviation maintenance environments, where accountability and traceability are essential operational requirements.

4.6. Summary Of the results

Based on the analysis presented in this chapter, the redesigned backend system demonstrates:

1. **Improved Reliability** Achieved through modular architecture, type-safe database access, and asynchronous processing.
2. **High Performance Stability** Consistently low API response times with minimal variance under repeated testing.
3. **Seamless SAP Integration** Reliable, non-blocking synchronization with zero data inconsistencies.
4. **Enhanced Observability** Real-time audit logging supporting traceability and operational transparency.

Collectively, these results confirm that the redesigned backend architecture effectively addresses the limitations of the legacy system and meets the operational demands of enterprise-scale aircraft maintenance systems.

5. Discussion and Conclusion

5.1. Discussion

The results of this study demonstrate that the redesigned backend architecture for the XOPS system provides significant improvements in data consistency, system performance, integration reliability, and operational observability. These findings not only confirm prior research on backend system optimization but also extend existing knowledge by validating these approaches within an enterprise aircraft maintenance environment integrated with ERP/SAP. In aviation maintenance logistics, backend reliability functions as a core operational infrastructure that supports the continuous flow of maintenance tasks, information verification, and regulatory documentation under strict time and safety constraints.

However, the primary contribution of this study should be understood as practice-oriented rather than broadly theoretical, as the work focuses on the design, implementation, and evaluation of a specific operational system configuration. Accordingly, the study contributes most directly to applied backend engineering in enterprise logistics environments, while offering empirical evidence that supports, rather than fundamentally advances, existing theoretical perspectives on service system reliability and information system performance.

The consistently low average API response time of 0.21 seconds supports earlier findings by Bhatt and Widatama, who reported that well-designed RESTful APIs improve scalability and service reliability. However, unlike previous studies that were evaluated primarily in experimental or cloud-centric environments, the results of this study were obtained from a real operational system with continuous transaction processing. This indicates that RESTful API optimization remains effective under enterprise operational constraints. In maintenance logistics operations, rapid response time is not only a performance indicator but a determinant of task continuity, as delays in information retrieval may interrupt inspection sequencing, maintenance verification, or component release decisions. In this regard, the study provides empirical validation that established backend optimization mechanisms remain operationally effective when deployed in a safety-critical, ERP-integrated maintenance environment.

The impact of Redis caching observed in this study aligns with the work of Kumar and Li, who demonstrated that in-memory caching stabilizes API performance by reducing repetitive database access. In the XOPS backend, Redis reduced authentication-related database queries by approximately 74% and achieved a cache hit ratio of 92%. The magnitude of this improvement is influenced by the high frequency of authentication requests in maintenance workflows, a factor that differentiates this study from prior research contexts. In aviation maintenance environments characterized by frequent task switching and high verification frequency, caching plays a critical role in preventing system access delays that could accumulate into operational bottlenecks. By stabilizing system responsiveness during

repetitive access patterns, caching supports uninterrupted service flow, which is essential for maintaining predictable maintenance turnaround time.

Asynchronous processing using RxJS played a crucial role in achieving reliable SAP integration. Anderson and Wong highlighted the benefits of reactive programming in managing non-blocking workflows; this study empirically validates those benefits in an enterprise integration scenario. The backend achieved 100% successful SAP synchronization with an average completion time of 0.83 seconds, demonstrating that asynchronous pipelines effectively isolate external system latency from core operational processes. This capability addresses a gap in prior studies, which rarely evaluated asynchronous processing in ERP-integrated operational systems.

This isolation mechanism is particularly critical in aviation maintenance logistics, where dependency on external enterprise systems such as SAP SE must not delay time-sensitive maintenance actions. By decoupling external synchronization from immediate operational workflows, asynchronous processing prevents propagation of integration delays into field-level activities, thereby preserving workflow continuity and minimizing service disruption risk.

From a database perspective, the application of DBSDLC principles, combined with normalization, indexing, partitioning, and Row-Level Security, resulted in query execution improvements of up to 38%. These results support Page's assertion that physical database design plays a critical role in enterprise performance optimization. Unlike studies that focus predominantly on application-layer enhancements, this research shows that database-centric optimization is equally important for achieving stable and predictable system behavior. Stable database performance is particularly important in maintenance logistics because maintenance records, component histories, and inspection outcomes must remain continuously accessible and internally consistent to support safe operational decision making.

The integration of Elasticsearch-based audit logging further distinguishes this study from previous work. While earlier research acknowledged the importance of audit logging, this study demonstrates its practical value in an aviation maintenance context by providing real-time observability and complete transactional traceability. This capability is essential for accountability and compliance in safety-critical operational environments.

In high-risk maintenance settings, observability is not merely a monitoring function but a safety assurance mechanism that enables rapid detection, diagnosis, and correction of operational anomalies. Complete traceability also reduces organizational risk exposure by ensuring that maintenance actions can be reconstructed, verified, and audited without ambiguity. This capability is particularly important when service disruption may have cascading safety and financial consequences, including delayed aircraft release, operational downtime, or regulatory non-compliance.

From a contribution perspective, the study therefore offers three distinct forms of value: practical engineering guidance for backend architecture design, empirical performance evidence from real operational deployment, and limited conceptual interpretation of backend mechanisms within logistics service systems. The theoretical implications remain interpretive and contextual rather than generative, as the study does not develop new formal models or test theory-driven causal hypotheses.

Despite these contributions, several limitations must be acknowledged. Performance testing was conducted under controlled operational conditions and did not simulate extreme peak workloads. SAP synchronization evaluation focused on transactional accuracy and latency rather than long-term behaviour under sustained high-volume integration. Additionally, the study was conducted within a single organizational and industry context, which may limit the generalizability of the findings.

Accordingly, the scalability and robustness of the redesigned backend should be interpreted as functionally demonstrated within observed operational conditions rather than fully validated under extreme or enterprise-wide load scenarios. The evaluation confirms improved responsiveness, stable system behaviour, and reliable integration performance within normal transactional ranges, but does not establish system behaviour under peak concurrency, long-duration stress exposure, or large-scale workload escalation. These conditions represent important boundary limits of the present evaluation

and define the operational scope within which performance claims are empirically supported. Comprehensive validation of peak-load resilience, long-term reliability, and infrastructure scaling capacity therefore remains an open area for future investigation.

Because the empirical evaluation is confined to one aviation maintenance organization and a single operational system architecture, the findings should be interpreted as context-sensitive rather than universally representative of all logistics service environments. Organizational structure, regulatory requirements, asset complexity, and maintenance workflow intensity vary substantially across industries, which may influence the magnitude of performance gains achievable through similar backend redesign strategies. Accordingly, direct generalization to other domains must be approached cautiously, particularly where operational risk profiles, system interoperability requirements, or information latency tolerances differ from those observed in aircraft maintenance.

Future research may address these limitations by conducting large-scale stress and endurance testing to evaluate backend behaviour under peak operational loads. Comparative evaluations of alternative integration architectures, such as message queues or event-streaming platforms, may provide deeper insight into optimal ERP integration strategies. Furthermore, advanced analytics applied to audit log data could enable predictive monitoring and automated anomaly detection. Such future investigations are particularly relevant for aviation maintenance logistics, where system resilience under peak operational pressure is directly linked to service continuity, safety assurance, and cost control.

Nevertheless, several conceptual and architectural insights from this study are transferable to other ERP-integrated logistics service systems. Mechanisms that stabilize information flow, such as low-latency API responses, reliable asynchronous integration, and comprehensive auditability, are broadly applicable to asset-intensive service environments that depend on real-time coordination between field operations and centralized planning systems. Examples include rail maintenance operations, where infrastructure inspection and repair activities must be synchronized with scheduling and traffic management systems, and maritime maintenance environments, where vessel servicing, port operations, and logistics planning rely on timely and consistent operational data exchange. In such contexts, backend architectures that reduce synchronization delay, prevent workflow blocking, and maintain traceable operational records can similarly improve service reliability, coordination efficiency, and operational transparency. Thus, while the empirical results remain domain-specific, the underlying design principles represent a transferable model for improving information-driven coordination in complex logistics service networks that integrate operational systems with enterprise platforms.

5.2. Managerial and Organizational Implications

The findings of this study carry several practical implications for managerial and organizational decision-making in aviation maintenance logistics. The improved reliability of backend information services enables maintenance managers to exercise tighter control over maintenance turnaround time by ensuring that operational data, job status updates, and verification records remain continuously accessible and synchronized across systems. This supports more predictable maintenance scheduling and reduces the risk of delays caused by information latency or system unavailability.

For IT architects and enterprise system planners, the study demonstrates the operational value of integrating asynchronous processing, distributed caching, and centralized auditability within ERP-connected service environments. These architectural mechanisms provide a structured approach to maintaining system responsiveness while preserving transactional consistency, particularly in contexts where field operations depend on continuous interaction with centralized enterprise platforms. The results therefore offer a practical reference for designing resilient integration layers that minimize workflow disruption when external systems experience latency or variability.

From a logistics coordination perspective, enhanced synchronization accuracy and system observability improve cross-functional visibility of maintenance activities. Logistics planners can rely on consistent real-time information when coordinating resource allocation, component handling, and maintenance sequencing across operational units. This reduces informational fragmentation and

supports more coherent planning across interconnected service processes.

At the organizational governance level, comprehensive audit logging strengthens compliance management and operational accountability. Decision-makers gain access to traceable and verifiable maintenance records, which support regulatory reporting, incident investigation, and performance monitoring. In safety-critical environments, such transparency contributes directly to risk control and informed decision-making.

5.3. Conclusion

This research successfully implemented the XOPS backend architecture as a robust service system by integrating PostgreSQL, Redis, RxJS, and Elasticsearch technologies. Evaluation results showed that improved backend reliability significantly contributed to service performance, as evidenced by a very low average API response time (0.21 seconds) and system stability under repeated workloads.

100% data synchronization accuracy with the SAP system ensured the integrity of logistics information, supporting more precise operational decision-making for maintenance management. Furthermore, the implementation of system observability through real-time audit logging increased accountability and transparency in service processes, essential requirements in an aviation safety environment. However, these findings should be interpreted as engineering validation under controlled operational conditions rather than definitive evidence of full enterprise-scale robustness. The study confirms functional scalability and improved service responsiveness within the evaluated workload range, but does not empirically establish performance under extreme demand, prolonged system stress, or large-scale distributed deployment. Consequently, claims regarding long-term reliability, peak-load resilience, and infrastructure scaling capacity should be regarded as prospective rather than fully demonstrated.

Overall, this research demonstrated that the modular, event-driven backend architecture not only met technical requirements but also ensured service continuity and laid the foundation for future digital service evolution. However, the empirical evaluation is primarily demonstrative and engineering-oriented, designed to validate system functionality and operational feasibility rather than to establish statistically generalizable performance effects. Performance improvements were assessed through descriptive comparison with the prior system configuration, without formal statistical hypothesis testing, variance modelling, or confidence interval estimation. Given operational data constraints and the real-world deployment context, the results should therefore be interpreted as engineering validation of system effectiveness rather than quasi-experimental causal evidence. Future work incorporating controlled benchmarking, repeated-measure comparisons, or before–after variance analysis would strengthen the statistical robustness and comparative credibility of the performance evaluation.

Acknowledgements

This work is supported by Universitas Multimedia Nusantara.

References

Abu Joma, M., Abu Lehyeh, S. M., & Albloush, A. (2024). The Effect of Job Enrichment on Organizational Citizenship Behaviors in Jordanian Industrial Companies. *Journal of Logistics, Informatics and Service Science*, *11*(3), 37–51. https://doi.org/10.33168/JLISS.2024.0303

Ahmad, S. (2022). Redis Performance and Scalability Review. *International Journal of Advanced Computer Science*, *12*(6), 55–60.

Ahmed, A., Aslan, H., & Fouad, K. (2025). Effective Integration of Database Security Tools into SDLC Phases: A Structured Framework. *Journal of Cybersecurity and Information Management*. https://doi.org/10.54216/jcim.160114

Anderson, L., & Wong, C. (2024). *Reactive Programming with RxJS 7*. Packt Publishing.

Aydemir, F., & Başçiftçi, F. (2025). Performance and Availability Analysis of API Design Techniques for API Gateways. *Arabian Journal for Science and Engineering*, *50*(15), 11485–11498. https://doi.org/10.1007/s13369-024-09474-9

de Oliveira, R. P., Lohmann, G., & Oliveira, A. V. M. (2022). A systematic review of the literature on air transport networks (1973-2021). *Journal of Air Transport Management*, *103*. https://doi.org/10.1016/j.jairtraman.2022.102248

Hartono, R. Z., & Wicaksono, D. H. (2023). Improving Audit Logs for Scalable Backend API. *Jurnal Teknik Informatika Indonesia*, *10*(1), 45–54.

Herdiyatmoko, H. F. (2022). Back-End System Design Based on REST API. *Jurnal Teknik Informatika (JUTIF)*, *5*(1), 50–57.

Hoffer, J. A., Ramesh, V., & Topi, H. (2023). *Modern Database Management*. Pearson.

Khan, J. W. (2023). SQL and NoSQL Database Software Architecture. *Big Data and Cognitive Computing*, *7*(2).

Kumar, R., & Li, V. (2022). Performance Evaluation of Redis for API Caching. *International Journal of Cloud Computing*, *18*(4), 221–230.

Kumar, R., & Singh, A. (2021). Evaluating ORM frameworks for enterprise applications. *Journal of Systems and Software*, *178*, 110995.

Lee, F., & Park, C. (2022). Integration of RESTful Web Services with ERP Systems. *IEEE Access*, *10*, 10001–10014.

Montoya-Murillo, D., Tavarez, M. M., Galván-Cruz, S., Zavala, A. E. M., & Rodríguez, F. Á. (2025). A review of SDLCs for big data analytics systems in the context of very small entities using the ISO/IEC 29110 standard-basic profile. *Int. Arab J. Inf. Technol.*, *22*, 194–215. https://doi.org/10.34028/iajit/22/1/15

Neeli, S. S. S. (2025). A Hands-On Guide to Data Integrity and Privacy for Database Administrators. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. https://doi.org/10.55041/ijsrem16443

Page, D. (2020). PostgreSQL as an enterprise database: A survey. *Open Source Database Journal*, *5*(2), 45–60.

Paudel, S. R., & Thapa, P. (2024). *Impact of Environmental, Social, Governance Factors on Consumers' Behavior in the Light of Digital Transformation*. https://doi.org/10.33168/SISD.2025.0106

Pratama, A., & Wella, W. (2024). Evaluation of Information Technology Services Using the Information Technology Infrastructure Library Framework. Jurnal Teknik Informatika dan Sistem Informasi, 10(2), 229-242.

Sachin Bhatt. (2024). Best Practices for Designing Scalable REST APIs in Cloud Environments. *Journal of Sustainable Solutions*, *1*(4), 48–71. https://doi.org/10.36676/j.sust.sol.v1.i4.26

Sharma, A. T. (2021). PostgreSQL features and innovations: Toward AI-ready databases. *Database Systems Journal*, *11*(3), 15–28.

Sherly, Wella. (2025). Optimization of Chatbot Model in Industry. International Journal on Informatics Visualization, Vol 9, No 6, 2698-2710. DOI: http://dx.doi.org/10.62527/joiv.9.6.3681

Shevchenko, O., Kuchapin, M., Dudar, Z., & Shirokopetleva, M. (2025). Enhancing Redis Cache Efficiency Based on Dynamic TTL and Adaptive Eviction Mechanism. *Proceedings of the Open Conference of Electrical, Electronic and Information Sciences, EStream*, *2025*, 1–6. https://doi.org/10.1109/eStream66938.2025.11016870

SITA. (2020). Air Transport IT insights. *Sita*, *261*(21), 17. https://www.sita.aero/resources/surveys-reports/air-transport-it-insights-2018/

Widatama, Y. B. A. (2024). Backend Infrastructure and Specifications Design Using OpenAPI and API-First. *Jurnal Indonesia Sosial Dan Teknologi*, *5*(8), 3710–3717.

Zhang, L., Chen, Z., & Xu, J. (2020). Middleware approaches to ERP–database integration. *Future Generation Computer Systems*, *113*, 465–476.

Zhang, Y., & Chen, C. (2023). Elasticsearch for Real-Time Logging Systems. *International Journal of Computer Applications*, *185*(7), 101–110.