 ISSN:2302-8149
e-ISSN:2540-9719

Performance Evaluation of a Laravel-based Internship Website using MySQL Master–Slave Replication

[1]Viona Mojang Pamungkas, [2]Galura Muhammad Suranegara*
[1,2]Program Studi Sistem Telekomunikasi, Universitas Pendidikan Indonesia
[1,2]Jl. Veteran No. 8, Nagri Kaler, Kabupaten Purwakarta, Indonesia
*e-mail: galurams@upi.edu

(***received*:** 9 February 2026, ***revised*:** 16 March 2026, ***accepted*:** 31 March 2026)

Abstract

Single database architectures often experience performance degradation as user load increases because all read and write operations are processed centrally, causing bottlenecks and limiting the scalability of web systems. To address this issue, this study implements database replication using a MySQL Master–Slave architecture combined with the separation of read and write operations in a containerized Laravel-based web system. A quantitative experimental method was applied by comparing the Single Database and Master–Slave Replication configurations implemented in a Docker environment on Windows Subsystem for Linux 2 (WSL2). Performance evaluation was conducted using Apache JMeter at three workload levels low (10 users), medium (50 users), and high (100 users) based on response time, throughput analysis, latency analysis, and error rate. The results show that the Master–Slave configuration provides more stable performance at medium to high workloads, with latency between 0.20 and 0.26 seconds and an error rate of 0% in all test scenarios. Additionally, the highest throughput of 2.01 requests per second was achieved at medium workload, indicating effective read-write separation and better workload distribution through database replication.

Keywords: database replication, latency analysis, read-write separation, throughput analysis, web system scalability

1 Introduction

Digital transformation has changed the operational capabilities of various industrial sectors, where the effectiveness of information systems is highly dependent on data processing efficiency and system scalability [1]. Digital transformation not only involves the adoption of technology, but also the strengthening of organizational capabilities that integrate business processes, human resources, and digital technology [2]. In this context, information system performance is an important aspect because it is directly related to service quality, user experience, and user satisfaction [3], [4].

Web-based information systems serve as a strategic tool in the provision of digital services, where system performance, ease of use, information quality, and service quality determine the effectiveness of the services provided to users [5], [6]. Web system development generally applies an agile approach to ensure flexibility and suitability with evolving operational needs, particularly in administrative and transaction systems [7]. However, in practice, many academic administration and internship management systems still rely on manual or semi-digital processes, making them inefficient and prone to data inconsistencies when handling large volumes of data [8], [9].

Performance issues in data-intensive web applications are generally related to the limitations of database architecture in handling increased workloads. Scalable database design and query optimization are necessary for the system to maintain performance when the number of users and transactions increases [10]. Empirical findings show that database optimization strategies play an important role in maintaining the stability and responsiveness of modern digital systems [11].

The distributed database approach through replication mechanisms is one solution to improve system scalability and availability while maintaining data consistency [12]. In addition, separating read and write operations on different database instances has been proven to improve system responsiveness and reduce the load on the main server [13]. To support this architecture, container technologies such as Docker provide a lightweight, flexible, and easily scalable deployment environment, thereby supporting system availability and reliability [14]. Empirical studies also show

 ISSN:2302-8149
e-ISSN:2540-9719

that a well-designed container architecture can improve performance stability under high connection load conditions [15].

Although database replication and containerization have been widely implemented, empirical studies integrating container-based MySQL master–slave replication with read–write separation mechanisms at the application level, particularly in Laravel-based web systems, remain limited. In particular, few studies have comprehensively evaluated system performance under various controlled workloads.

Based on these issues, this study aims to evaluate the performance and stability of Laravel-based web systems that implement MySQL masterslave replication architecture in a Docker container environment. The evaluation was conducted at low, medium, and high workload levels by analyzing system response, data consistency, and error rates. The results of this study are expected to serve as a reference in the design of scalable and highly available web-based information systems.

2 Literature Review

The development of increasingly complex web-based information systems requires a backend architecture that is capable of handling high access loads in a stable and efficient manner. Several studies confirm that traditional database architectures with a single database are still widely used due to their ease of implementation and maintenance, but they have limitations in terms of scalability and service availability when the number of users increases simultaneously [10], [12]. These limitations have the potential to reduce system performance and increase the risk of service failure in applications with high access intensity.

As a solution, database replication is widely used to distribute workloads and improve system reliability. [8], [16] show that the implementation of database replication with a master–slave scheme can reduce the load on the main server and speed up the data retrieval process in web-based applications. However, these studies generally still focus on general performance comparisons, without evaluating overall system stability under varying access load conditions. In addition, the load characteristics tested do not fully represent the actual access patterns in management information systems.

In modern applications developed using the Laravel framework, several recent studies have begun to examine the integration of distributed database architecture to improve service availability. [11], [17] report that the use of database replication in Laravel applications can maintain data consistency and improve system reliability. However, these studies have not specifically compared the performance between a single database and master–slave replication using measurable and standardized load testing scenarios. The performance evaluations conducted are also limited to response times, thus failing to provide a comprehensive picture of system stability.

Several other studies emphasize the importance of load-based performance testing to assess system readiness in real operational conditions. [18], [19] show that the use of testing tools such as Apache JMeter is effective in simulating simultaneous user access and identifying system weaknesses. However, these studies have not directly linked the test results to differences in database architecture, particularly in Laravel applications that have periodic report access patterns with a predominance of read operations. In fact, high read load characteristics greatly affect the effectiveness of database replication implementation [20].

Based on this review, it can be concluded that there are still gaps in research related to comparative analysis of performance and stability between single database architecture and MySQL master–slave replication in Laravel-based applications. Therefore, this article focuses on testing both architectures using several levels of access load, namely low, medium, and high, with evaluation indicators in the form of latency and request error rates. This approach is expected to provide empirical contributions in determining a more effective and reliable database architecture to support scalable web-based information systems.

3 Research Method

This study uses a quantitative experimental approach designed to evaluate the effectiveness and efficiency of distributed database architecture in Laravel-based web systems. This approach focuses

 ISSN:2302-8149
e-ISSN:2540-9719

on performance testing through empirical measurements of two different configurations, namely a single database and a MySQL Master–Slave replication architecture consisting of one master node and two slave nodes. All tests were conducted in a controlled environment using Docker Engine running on the Windows Subsystem for Linux 2 (WSL2) Server operating system to ensure consistency and reproducibility of the experimental settings. The methodological design adopts a quantitative experimental framework that relies on controlled performance measurements, including response time, throughput, and latency, to evaluate system behavior under different database configurations, as commonly applied in performance evaluation studies of distributed and microservices-based information systems [21]. The overall research methodology process is illustrated in Figure 1.

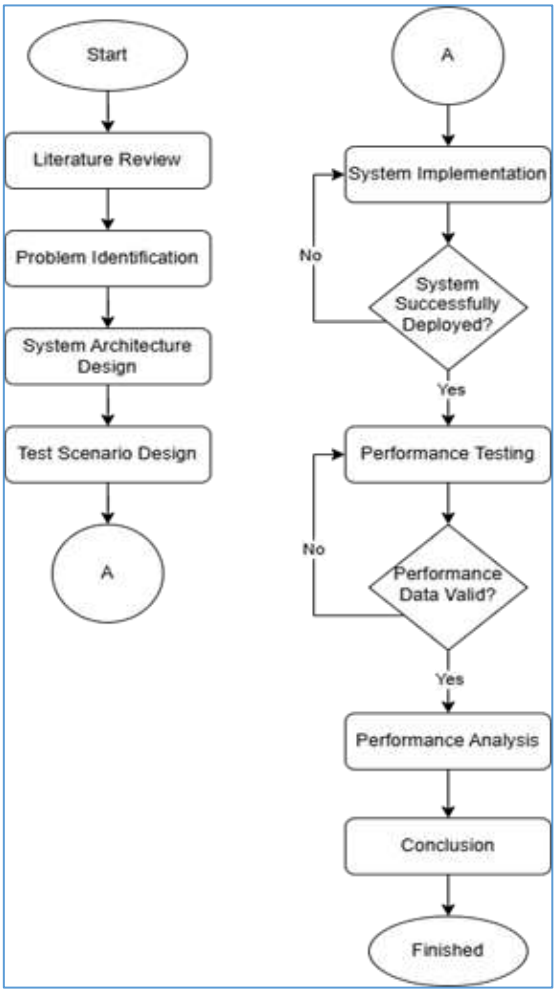

Figure 1 Research methodology flowchart

3.1 Literature Riview

The literature study stage was conducted to obtain the theoretical and technical basis that serves as a reference in the design and implementation of a database replication system using a MySQL master–slave architecture in a Laravel-based application. This activity included the process of searching and analyzing relevant previous studies, especially those discussing the performance of replication systems, data synchronization, and the application of containerization using Docker in a distributed environment. The results of the literature study were used to identify commonly applied approaches, the advantages and limitations of each method, and aspects that had not been widely evaluated in previous studies. These findings then became the basis for formulating the research focus, determining the testing scenario, and selecting relevant evaluation parameters, so that the research conducted had a clear direction and was able to contribute to the development of a reliable and scalable web-based information system.

3.2 Problem Identification

 ISSN:2302-8149
e-ISSN:2540-9719

Based on the results of a literature review, it can be identified that most Laravel-based web systems still use a single database architecture that is prone to performance bottlenecks when user load and data volume increase. Meanwhile, although MySQL master–slave replication has been proven to improve availability and read load distribution, quantitative performance evaluations in the context of Laravel are still rare. This study focuses on filling this gap by conducting a comparative analysis between Single Database and Master–Slave Replication configurations to assess their impact on response time, throughput, and error rate in Laravel-based internship systems.

3.3 System Architectural Design

This study implements a horizontally scalable system architecture designed to support high availability and efficient workload distribution in a web-based application environment. All database components are deployed as isolated Docker containers running on a Windows Subsystem for Linux 2 (WSL2) host to ensure environment consistency, service isolation, and data persistence through volume mapping mechanisms.

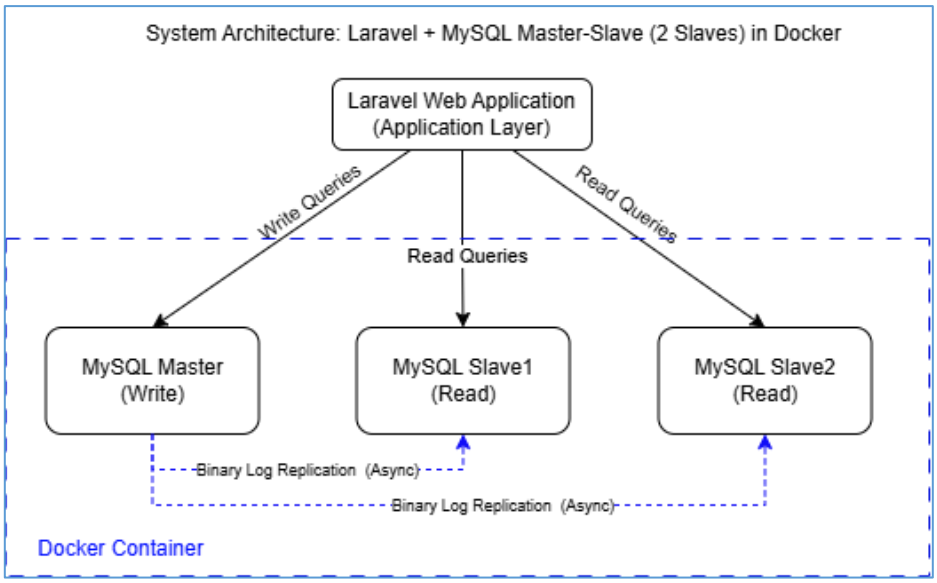

Figure 2 System architecture of master–slave database replication

Figure 2 illustrates the system architecture, which consists of a Laravel-based web application at the application layer and a MySQL database cluster at the data layer. The database layer follows a master–slave replication model, where a single MySQL master node handles all write operations, while two MySQL slave nodes are dedicated to serving read requests.

3.4 Test Scenario Design and System Implementation

At this stage, a system performance testing scenario is designed to evaluate the effectiveness of the proposed distributed database architecture. Apache JMeter is used as a load testing tool to measure the system's ability to handle concurrent user requests under different workload conditions. The testing focuses on the monthly report endpoint, which represents a read-intensive operation and is one of the most frequently accessed features in the application. The test scenarios were categorized into three workload levels: low, medium, and high. Each workload scenario was executed independently under controlled conditions. The detailed configuration of each workload level is presented in Table 1.

Table 1 Load testing scenario

Load Category	Threads (Users)	Time (Seconds)	Loop	Req
Low	10	10	2	20
Medium	50	30	5	100

 ISSN:2302-8149
e-ISSN:2540-9719

High	100	30	5	500

To ensure that the results are consistent and there are no momentary deviations, each scenario is repeated five times. Furthermore, key metrics, such as average response time, throughput, and request success rate, are used to analyze the test results. To gain insight into the impact of horizontal replication on system efficiency and stability, a single database architecture is compared with Master–Slave replication. To support research reproducibility, the specifications of the implementation environment and supporting software are detailed in Table 2.

Table 2 Environment and software specifications

Component	Software/Tool	Version
Operating System	Windows 11 (WSL 2)	-
Container Engine	Docker Engine	29.1.3
Orchestration	Docker Compose	v2.40.3-desktop
Application	Laravel Framework	10.48.28
Database	MySQL	8.0
Testing Tool	Apache JMeter	5.6.3

Through this structured testing design, performance evaluation was conducted in a controlled and systematic manner to ensure that the obtained results accurately reflected the actual system performance. This stage also adheres to the principles of reproducibility and data consistency, enabling the experimental results to serve as a reliable basis for empirical evaluation in research on distributed system architectures.

3.5 Performance Testing and Analysis

Performance testing was conducted on the monthly report endpoint to evaluate the effectiveness of the system's read operations. The primary performance metrics observed were response time, throughput, and error rate, which collectively represent the system's performance and stability. All test result data were collected using the JMeter Summary Report and exported in CSV format to facilitate systematic processing in the subsequent analysis stage.

4 Results and Analysis

Performance evaluation was conducted to compare the Single Database and MySQL Master–Slave Replication architectures at low (10 users), medium (50 users), and high (100 users) load levels. The parameters analyzed included response time, throughput, latency, and error rate to assess the scalability and stability of the system.

In the Single Database configuration, all read and write operations are processed by a single MySQL server without a load distribution mechanism. Based on the test results in Table 3, the system is capable of providing a fast response time of 626 ms under low load. This shows that a centralized architecture is still effective when the number of simultaneous users is relatively small.

Table 3 Single database performance results

Threads (Users)	Response Time (ms)	Throughput (Request/Sec)	Latency (s)	Error Rate (%)
10	626 ms	1.82 requests/sec	0.22 s	0.0%
50	19.196 ms	1.85 requests/sec	0.23 s	0.0%
100	36.899 ms	2.11 requests/sec	0.26 s	0.0%

However, as the number of users increases, there is a very significant increase in response time, reaching 19.196 ms under medium load and 36.899 ms under high load. This condition indicates a performance bottleneck due to centralized processing, where all read and write requests compete for the same computing and I/O resources. This finding is in line with research by Pohanka and Pechanec, which states that centralized database architectures tend to experience drastic performance degradation under high concurrency conditions. Although throughput values experience a slight

 ISSN:2302-8149
e-ISSN:2540-9719

increase, the spike in response time indicates the system's limitations in handling simultaneous workloads efficiently [20].

The MySQL Master–Slave Replication architecture implements workload separation by directing write operations to the master node and read operations to the two slave nodes. Based on the test results in Table 4, the response time under low load is slightly higher than that of a single database due to replication and data synchronization overhead.

Table 4 Master–slave replication performance results

Threads (Users)	Response Time	Throughput (Request/Sec)	Latency (s)	Error Rate (%)
10	959 ms	1.66 requests/sec	0.20 s	0.0%
50	17.049 ms	2.01 requests/sec	0.25 s	0.0%
100	40.444 ms	1.96 requests/sec	0.24 s	0.0%

However, under medium load, the Master–Slave configuration showed the best performance with a response time of 17.049 ms and the highest throughput of 2.01 req/sec. These results indicate that separating read and write operations effectively reduces the load on the master node and improves data processing efficiency. These findings are consistent with the concept of read–write separation proposed by Ramanathan et al., which states that distributing read requests to multiple nodes can improve system scalability and responsiveness [13].

Under high load conditions, response time increased to 40.444 ms; however, the system remained stable with an error rate of 0%, indicating that the replication mechanism was able to maintain system availability and reliability even under maximum workload. These results are consistent with the findings of Georgiou et al. and Wibowo et al., which emphasize that database replication plays a crucial role in improving the stability and availability of distributed systems [12], [17]. This performance trend is further reinforced by the visualization results shown in Figures 3, 4, and 5. Figure 3 shows that the Single Database architecture provides better response times under low load conditions, but experiences a drastic decline when the workload increases to medium and high levels due to centralized processing. In contrast, the Master–Slave architecture shows more stable and scalable performance by distributing read requests to multiple nodes, which allows for better handling of read-intensive workloads commonly found in web-based information systems.

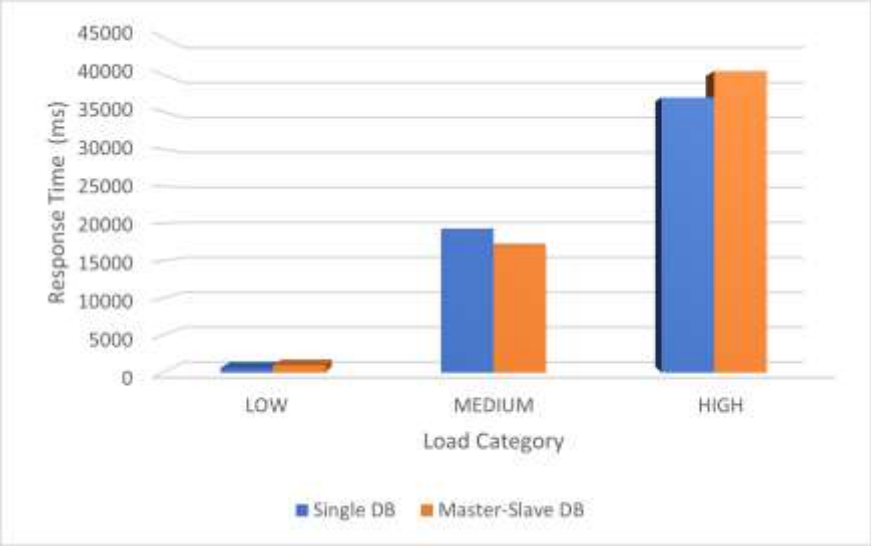

Figure 3 Visual graph comparing response time

The comparison of response times shown in Figure 3 indicates that the Single Database architecture performs better under low load conditions due to its simpler architecture and absence of replication overhead. However, as the number of concurrent user connections increases, the Master–Slave architecture shows a more stable performance trend, especially at medium load levels. These results extend previous studies by providing empirical evidence that replication-based architectures

 ISSN:2302-8149
e-ISSN:2540-9719

are not only beneficial in large-scale systems but also improve performance consistency in medium-scale applications.

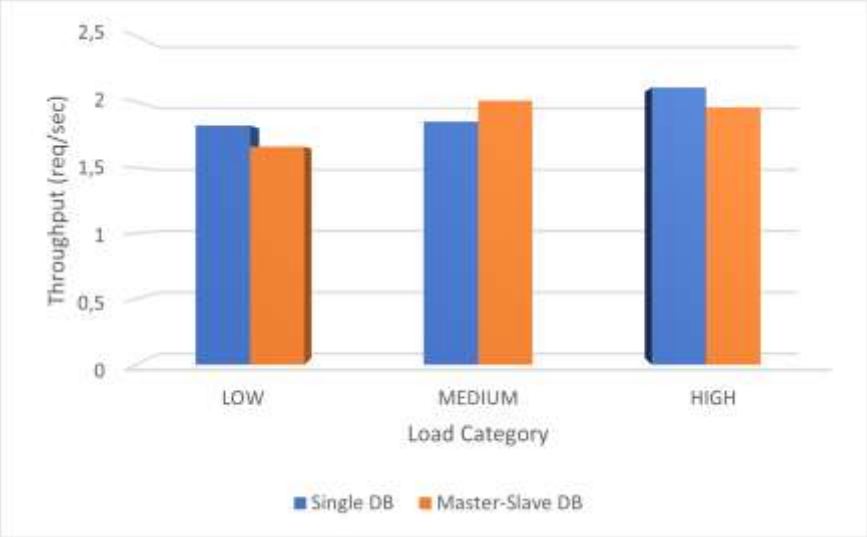

Figure 4 Visual graph comparing throughput

Figure 4 shows that at medium loads, the Master–Slave architecture produces higher throughput than the Single Database. These results support the findings of Ramanathan et al., who stated that separating read loads can improve the efficiency of parallel request processing. At high loads, both architectures show relatively comparable throughput values, indicating that the system has approached its optimal capacity limit [13].

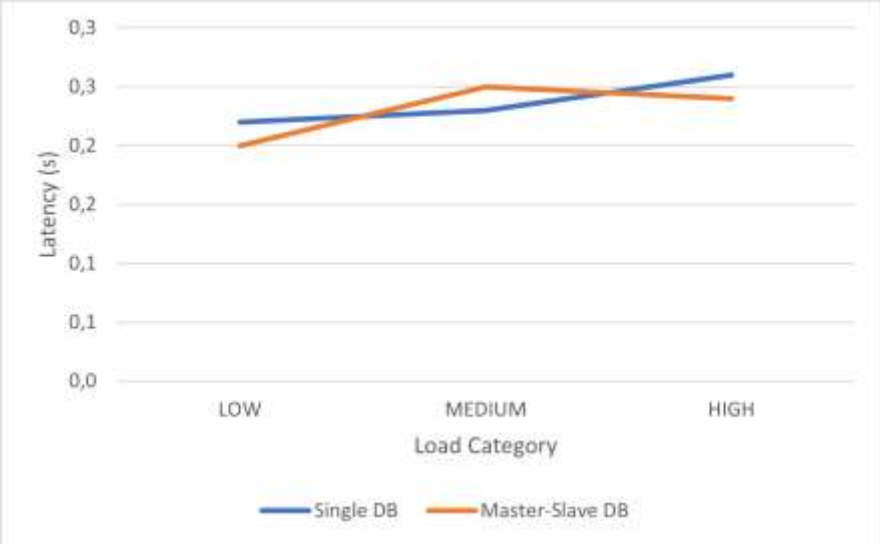

Figure 5 Visual graph comparing latency

The latency trends shown in Figure 5 indicate that both architectures are capable of maintaining relatively low and stable latency. However, the Master–Slave architecture consistently has slightly lower latency under medium and high loads. This shows that the use of asynchronous replication and Docker containerization does not have a significant negative impact on network performance, as also reported by Gkamas et al. and Haryoto et al. in their research on containerized database systems[19], [22].

Overall, the results of this study show that although the Single Database architecture is still adequate for low-load environments, the implementation of MySQL Master–Slave Replication combined with a read–write splitting mechanism at the application level provides increased scalability, stability, and performance efficiency. The uniqueness of this study lies in its comprehensive performance evaluation of a Laravel-based web application implemented end-to-end

 ISSN:2302-8149
e-ISSN:2540-9719

in a Docker environment, thereby expanding on previous studies that generally focused only on testing databases or infrastructure separately.

5 Conclusion

This study concludes that the implementation of MySQL Master–Slave distributed database architecture in a Docker environment can improve the performance and stability of Laravel-based internship management systems, especially under medium to high user loads. Test results show that the mechanism of separating read and write operations and distributing read requests to several slave nodes is effective in increasing throughput, stabilizing response times, and maintaining low latency without causing system errors. The results confirm that the MySQL Master–Slave replication architecture improves system scalability and stability compared with the single database configuration, particularly under medium and high workloads.

References

[1] Sasmoko, Y. Indrianti, S. R. Manalu, and J. Danaristo, "*Analyzing Database Optimization Strategies in Laravel for an Enhanced Learning Management*," *Procedia Comput. SCI.*, Vol. 245, No. C, pp. 799–804, 2024, DOI: 10.1016/j.procs.2024.10.306.

[2] A. F. Rizana, I. I. Wiratmadja, and M. Akbar, "*Exploring Capabilities for Digital Transformation in the Business Context: Insight from a Systematic Literature Review*," *Sustain.*, Vol. 17, No. 9, 2025, DOI: 10.3390/su17094222.

[3] H. Fa'iqoh and D. F. Suyatno, "Evaluasi *User Experience* Aplikasi Indonesia Pustaka Nasional (IPUSNAS) menggunakan Metode *Heart Metrics*," *J. Emerg. Inf. Syst. Bus. Intell.*, Vol. 05, No. 02, pp. 126–135, 2024, DOI: 10.26740/jeisbi.v5i2.60419.

[4] L. Setiawati, A. Hadiapurwa, B. Santoso, H. Nugraha, and P. Rusli, "*Importance-Performance Analysis of SINERGI UPI Website User Satisfaction*," *J. Edutech Undiksha*, Vol. 12, No. 1, pp. 20–27, 2024, DOI: 10.23887/jeu.v12i1.64450.

[5] F. Islamiah and R. Wijaya, "Penilaian Kepuasan Pengguna *Website* Sistem Informasi Akademik menggunakan Metode *Website Quality*," *METIK (Media Teknologi Informasi dan Komputer Jurnal).*, Vol. 6, No. 2, pp. 133–139, 2022, DOI: 10.47002/metik.v6i2.381.

[6] S. Muna, Nurdin, and Taufiq, "*Comparative Analysis of State Universities on Website Performance in Aceh using the PIECES Method*," *JITE (Journal of Informatics Telecommunication Engineering).*, Vol. 7, No. July, pp. 71–83, 2023, DOI: 10.31289/jite.v7i1.9167.

[7] S. A. Fadillah, N. Chandra, and C. Rivatunisa, "Implementasi *Agile Scrum* pada Pembuatan *Website* Sistem Informasi Manajemen Kuliner," *Decod. J. Pendidik. Teknol. Inf.*, Vol. 4, No. 1, pp. 301–315, 2024, DOI: 10.51454/decode.v4i1.357.

[8] L. M. Diana and E. M. Stefany, "Pengembangan *Website* Administrasi Skripsi Program Studi Pendidikan Informatika," *Decod. J. Pendidik. Teknol. Inf. ISSN*, Vol. 3, No. 2, pp. 229–235, 2023, DOI: 10.51454/decode.v3i2.167.

[9] A. G. Karyn, P. Pahrudin, and A. A. Khair, "*Design of the Management Information System for Interns at the Communication and Information Service of East Kalimantan Province*," *Int. J. Inf. Technol.*, Vol. 9, No. 158, pp. 93–106, 2025, DOI: 10.30645/ijistech.v9i1.401.

[10] E. C. Foster and S. V. Godbole, *Database Systems: A Pragmatic Approach, 3rd edition*. Boca Raton, FL, USA: Auerbach Publications, 2022.

[11] E. Firmansyah, H. Firdaus, and Samidi, "Optimasi Performa *Query* Subsidi Debitur dengan Index and *Table Partition, Subquery and Indexing*, dan *Parallel Query Execution*," *J. Pendidik. dan Teknol. Indones.*, Vol. 5, No. 6, pp. 1769–1785, 2025, DOI: 10.52436/1.jpti.824.

[12] M. A. Georgiou, M. Panayiotou, M. Sirivianos, and H. Herodotou, "*Attaining Workload Scalability and Strong Consistency for Replicated Databases with Hihooi*," No. June, 2021, DOI: 10.1145/3448016.3452746.

[13] S. Ramanathan, G. Gautam, V. Srinivasan, and P. Parag, "*Latency-Redundancy Tradeoff in Distributed Read-Write Systems*," *2022 14th Int. Conf. Commun. Syst. NETworkS, COMSNETS 2022*, pp. 172–180, 2022, DOI: 10.1109/COMSNETS53615.2022.9668414.

[14] S. P. Kane and K. Mathhias, *Docker: Up & Running: Shipping Reliable Containers in*

Production. O'Reilly Media, 2023.

[15] M. R. Fachrudin and A. R. Muslikh, "*Optimization of Application Deployment Architecture in Container Orchestration*," *J. Appl. Informatics Comput.*, Vol. 9, No. 2, pp. 383–392, 2025, DOI: 10.30871/jaic.v9i2.8972.

[16] T. Abdillah and I. P. E. Prismana, "Analisis Performansi *Web Server* menggunakan *Load Balancing* pada *Virtualisasi Docker Container*," *J. Informatics Comput. SCI.*, Vol. 03, No. 04, pp. 526–533, 2022, DOI: 10.26740/jinacs.v3n04.p526-533.

[17] D. K. Wibowo, A. Darmawan, and D. A. Nawangnugraeni, "*A Comparative Study of Multi-Master Replication of NOSQL Database Server with Varying Data Formats*," *J. Tek. Inform.*, Vol. 6, No. 1, pp. 411–418, 2025, DOI: 10.52436/1.jutif.2025.6.1.4371.

[18] I. Sholihah and C. Darujati, "Sistem Replikasi basis Data berdasarkan *MySQL* menggunakan *Container Docker* [*Database Replication System based on MySQL using Docker Containers*]," *Maj. Ilm. Teknol. Elektro*, Vol. 21, No. 2, p. 209, 2022, DOI: 10.24843/mite.2022.v21i02.p08.

[19] I. S. Haryoto, G. Firmansyah, B. Tjahjono, and A. M. Widodo, "*Evaluation and Optimization of MySQL Master-Slave Performance with Quantitative Methods on the Database of Psychology Test Applicants SIM PT XYZ at Polda Metro Jaya*," *LOCUS*, Vol. 4, No. 9, pp. 8896–8905, 2025, DOI: 10.58344/locus.v4i9.4685.

[20] T. Pohanka and V. Pechanec, "*Evaluation of Replication Mechanisms on Selected Database Systems*," *ISPRS Int. J. Geo-Information*, Vol. 9, No. 4, 2020, DOI: 10.3390/ijgi9040249.

[21] R. Wati, N. Andriyani, T. Susilowati, S. Informasi, F. Teknologi, and I. B. Nusantara, "Rancang Bangun Sistem Informasi Akademik berbasis *Microservices*," *J. Artif. Intell. Digit. Bus.*, Vol. 4, No. 4, pp. 4906–4912, 2025, DOI: 10.31004/riggs.v4i4.4006.

[22] T. Gkamas, V. Karaiskos, and S. Kontogiannis, "*Performance Evaluation of Distributed Database Strategies using Docker as a Service for Industrial IoT Data: Application to Industry 4.0*," *Inf.*, Vol. 13, No. 4, 2022, DOI: 10.3390/info13040190.