Journal of Business, Social and Technology
Volume 7, Issue 2, 482-495
e_ISSN: 2807-6362
https://bustechno.polteksci.ac.id/index.php/jbt/index
DOI: doi.org/10.59261/jbt.v7i2.654

Implementation of Event-Driven Architecture using NATS JetStream for a Large-Scale Online Exam Answer Collection System

***Dery Ciputra Ma'soem[1]**
STMIK IM, Indonesia

Novi Rukhviyanti[2]
STMIK IM, Indonesia

***Corresponding author:**
Dery Ciputra Ma'soem, STMIK IM, Indonesia.
✉ radendiddyjuned@gmail.com

Article Info:

Article history:
Received: May 04, 2026
Revised: May 18, 2026
Accepted: May 20, 2026

Keywords:
asynchronous computing; event-driven architecture; load testing; nats jetstream; online exam

Abstract

Background: Online examination systems—even at the prototype and institutional levels—face serious performance challenges under synchronized spike conditions. This study, conducted on a prototype (*Chilaedu*) with 300 synthetic users, demonstrates that conventional synchronous (*request-response*) architecture has thread-capacity limitations that trigger gateway timeout failures, thereby undermining overall assessment system reliability.

Objective: This research designs and evaluates the architectural transition toward an asynchronous *Event-Driven Architecture* (EDA) approach using the Go runtime and the *NATS JetStream* message broker on the *Chilaedu* e-learning platform prototype.

Methods: System performance was evaluated using a load-testing scenario based on a custom testing tool developed with Go and Python to simulate synchronized spike conditions at concurrency levels of 10, 25, 50, and 100, with 300 synthetic user entities per scenario, in a single-server isolated prototype environment.

Results: The test results show that the asynchronous architecture successfully maintained a throughput of 2,209 requests per second with an HTTP error rate of 0.00%. The system operated efficiently with an average memory consumption of 70 MB and a peak CPU utilization of 18.03%. Latency at the 99th percentile (*p99*) was recorded at 79.02 ms, which is below the feasibility threshold of 100 ms, thereby preventing user interface freezes (*browser freezes*). A comparison with the synchronous monolithic approach documented in previous research shows a significant increase in load-handling capacity.

Conclusion: The *NATS JetStream*-based EDA architecture proves effective as a solution for improving the reliability and efficiency of large-scale online exam answer collection infrastructure.

To cite this article: Dery Ciputra Ma'some, N., & Rukhviyanti, N. (2026). Implementation of event-driven architecture using NATS JetStream for a large-scale online exam answer collection system. *Journal of Business, Social and Technology*, 7(2), 482–495. https://doi.org/10.59261/jbt.v7i2.654

INTRODUCTION

Digital transformation in the education sector has driven the adoption of online examination systems as an essential infrastructure component in the learning outcome evaluation process (Oktaviane & Rukhviyanti, 2026; Paredes et al., 2023; Sahni et al., 2025; Shard et al., 2024). As e-learning platform adoption expands, back-end server reliability and the capacity to sustain large concurrent user loads become dominant technical imperatives—particularly under the peak-submission scenarios inherent in online examinations (Blinowski et al., 2022; Kleppmann, 2017).

Technical problems that often arise in the implementation of online exams are related to request spikes that occur simultaneously when students submit answers at nearly the same time.

This phenomenon can be categorized as a synchronized spike, which has the potential to cause system performance degradation and even service failure (Arief et al., 2024). Under these conditions, the server system experiences high pressure due to the accumulation of simultaneous requests, which can lead to processing delays, data loss, or connection failures. The impact of such failures is not merely technical but also affects the validity of exam results and the overall credibility of the academic evaluation process. Empirically, such conditions produce tangible failure modes: (1) HTTP 504 Gateway Timeout responses returned before student answers are persisted; (2) lost submissions in which the client receives no error, but database writes fail because of thread pool exhaustion; (3) duplicate submissions triggered by user retries after apparent failures; and (4) row-level locking deadlocks in the exam session table caused by concurrent write contention—each directly compromising exam data integrity (Abdelsalam et al., 2023; Kleppmann, 2017).

System reliability in the context of online exams is closely related to user trust (Njuguna, 2022). System failure, especially during the final answer submission phase, can create uncertainty regarding the storage status of data sent by participants. This can potentially lead to doubts about the integrity of exam results, especially if the system cannot provide deterministic guarantees of data consistency. Therefore, reliability and fault tolerance become important elements in the design of online exam system architecture (Abdelsalam et al., 2023). In practice, the synchronous request-response approach commonly used in conventional HTTP-based architectures often faces limitations in handling high-concurrency workloads, especially when each request must wait for the database write process to complete before the connection is closed.

This synchronous approach tends to create a bottleneck at the database layer because of the high level of access contention among various processes running in parallel. Within a distributed systems framework, this condition can significantly increase latency and the likelihood of request failures, such as timeouts or connection drops (Kleppmann, 2017).

A number of empirical studies show that synchronous architecture, particularly in monolithic form, experiences significant performance degradation when faced with a large number of users compared to event-driven approaches (Cabane & Farias, 2024). Besides affecting technical aspects, system instability in online exam contexts can also degrade the quality of the learning experience and potentially affect student motivation to participate in the evaluation process (Elumalai et al., 2021; Hasbi et al., 2020).

Efforts to increase system capacity through vertical scaling often face limitations, both in terms of cost and implementation flexibility, especially in educational institutions with limited resources. Therefore, an alternative approach is needed that can improve scalability without depending significantly on increased hardware specifications. One widely studied approach is the implementation of Event-Driven Architecture (EDA), which allows data processing to be performed asynchronously through a message-passing mechanism. In this architecture, client requests are not processed directly to completion but are instead delegated to a message queue to be processed by separate workers.

This event-driven approach allows the system to reduce dependence on blocking operations and minimize contention for shared resources such as databases. Previous studies have shown that the use of message brokers and background workers can improve processing efficiency and reduce average latency in high-load systems (Cox-Buday, 2017). Furthermore, stream-processing approaches in microservice architectures have also been shown to achieve near-linear scalability with the addition of computing resources (Henning & Hasselbring, 2024). In implementation contexts, various message broker technologies such as Apache Kafka and RabbitMQ have been widely used to support large-scale data processing, particularly in high-throughput scenarios and complex data-stream processing (Goel, 2024; Lai & Huynh, 2022).

On the other hand, NATS JetStream has emerged as an alternative message broker designed with a focus on low latency and efficient publish-subscribe and request-reply communication, as documented in its official technical specifications (NATS, 2023). These characteristics make it relevant for latency-sensitive scenarios, particularly in exam answer submission processes that require fast and consistent responses, especially when compared to other message brokers optimized for high throughput, such as Apache Kafka, or routing flexibility, such as RabbitMQ (Goel, 2024; Lai & Huynh, 2022). However, most existing research still focuses

on message broker implementations in large-scale distributed environments, such as Kubernetes-based clusters, and has not specifically evaluated their performance on limited infrastructure with single-server configurations.

A structured comparison clarifies the gap: Arief (2024) tested lightweight Kubernetes (K3s) clusters for online exam reliability, demonstrating multi-node scalability but requiring complex orchestration absent from most Indonesian schools; Cabane (2024) evaluated EDA using Apache Kafka in distributed cloud environments without addressing single-server deployments; Blinowski (2022) benchmarked monolithic versus microservice REST architectures without asynchronous queuing strategies; and Goel (2024) compared Apache Kafka, RabbitMQ, and NATS in generic workloads rather than education-specific exam-submission scenarios. No prior work has examined NATS JetStream with the Go runtime on single-server infrastructure for the burst-submission pattern of online examinations—the gap this study addresses.

This research gap becomes significant in the context of implementation in secondary education institutions, particularly in Indonesia, which generally have limited infrastructure resources. To date, few studies have explicitly evaluated the effectiveness of integrating the Go programming language with NATS JetStream in handling concurrent request spikes in single-server-based online exam systems. Yet, this approach holds the potential to provide a more affordable solution without sacrificing system performance and reliability.

Based on this background, this study aims to implement and evaluate an event-driven architecture using Go and NATS JetStream in the context of online exam answer collection. The main focus of the research is to measure the system's ability to maintain throughput and minimize error rates under synchronized spike conditions. Specifically, this study seeks to answer the following question: "Does the implementation of an Event-Driven Architecture based on NATS JetStream on a single-server infrastructure improve system reliability and performance in handling simultaneous online exam answer submission loads?"

METHOD

This research used the Design Science Research Methodology (DSRM) framework, which focused on the development and evaluation of software engineering artifacts in the context of e-learning systems. The selection of DSRM was based on its ability to bridge practical needs and scientific contributions through the development of technology solutions that could be empirically tested (Vom Brocke et al., 2020). In the context of this research, the artifact developed was an online exam system architecture based on Event-Driven Architecture (EDA), designed to overcome the problem of synchronized spikes. Unlike traditional software engineering approaches oriented toward development processes, DSRM emphasized the validation of solution effectiveness through measurable quantitative testing.

Methodologically, this research followed three main stages in the DSRM cycle: (1) problem identification, which focused on the analysis of online exam loads that caused system performance degradation; (2) design and development, which consisted of designing an asynchronous EDA-based architecture with the integration of system components such as the Go programming language, NATS JetStream message broker, and Redis as an in-memory cache layer; and (3) evaluation, which was conducted through load testing to quantitatively measure system performance based on specified metrics.

This research was conducted over a three-month period, from January to March 2026. The initial stage of observation and system requirements analysis was carried out at SMK Kiansantang Bandung as a representative case study. This institution was chosen because it had the characteristic of using an online exam system with a high number of simultaneous participants, making it potentially susceptible to bottleneck conditions in server infrastructure. To maintain the internal validity of the research, the system development and performance testing processes were not conducted directly in the production environment but rather on an isolated infrastructure based on a Virtual Private Server (VPS) using Microsoft Hyper-V virtualization technology.

This approach aimed to control external variables such as public network fluctuations and to ensure consistency in system performance measurements.

The object of this research was the Chilaedu e-learning platform, which had been modified

to implement an EDA-based architecture. Meanwhile, the subject of the research in the experimental context did not directly involve human users but instead used 300 synthetic user entities designed to simulate simultaneous exam answer submission behavior. The use of synthetic users was chosen to ensure experiment reproducibility and eliminate uncontrolled variables such as human response time or device variation. Furthermore, this approach also considered research ethics by avoiding potential disruptions to actual academic processes.

All experiments were conducted in an isolated virtual computing environment. The server specifications used are presented in Table 1 as part of research transparency and reproducibility.

Table 1. Test Server Infrastructure Specifications

Component	Detailed Specifications
Processing Unit (CPU)	Intel Xeon® 4314 @ 2.40GHz (4 Cores)
Memory (RAM)	8 GB
Storage	NVMe/SSD (I/O Speed Avg. 1.9 GB/s)
Operating System	Debian GNU/Linux 13 (trixie)
Kernel	6.12.74+deb13+1-amd64
Connectivity	1 Gbps (Telkom Indonesia AS7713)

The developed system architecture adopted an Event-Driven Architecture (EDA) approach with a clear separation between the main response path (hot path) and background processing. The Go programming language was used as the implementation foundation because of its ability to handle lightweight concurrency through an efficient goroutine mechanism for I/O-bound workloads. In this architecture, the API server acted as the main entry point for user requests, while heavy data-processing tasks were delegated to the asynchronous queue system.

The main components of the architecture included: (1) an API server that handled request validation and initial responses; (2) a NATS JetStream-based message broker that functioned as an asynchronous communication intermediary as well as persistent message queue storage; (3) a background worker tasked with processing data separately from the main path; and (4) a storage system consisting of a PostgreSQL relational database and Redis as an in-memory cache. This approach allowed the system to minimize waiting time on the client side while increasing resilience to load spikes.

The exam answer submission process flow was designed asynchronously to avoid dependence on synchronous operations that could potentially cause system bottlenecks. When a user submitted an answer, the system first performed authentication and business-rule validation. After successful validation, the data was not processed directly into the database but was instead sent to the queue system as an event. The API then provided an initial response to the client without waiting for the storage process to complete. Subsequently, the background worker independently retrieved messages from the queue and performed further processing, including data storage and result calculation. This mechanism allowed the system to maintain responsiveness even during simultaneous request spikes.

This design approach also adopted the event-carrying state transfer principle, in which the data structure sent through the queue was kept minimal and efficient to reduce communication overhead (Oliveira Rocha, 2021). Additionally, the queue system was configured with an acknowledgment mechanism to ensure message-processing reliability, including the ability to support redelivery in cases of failure during the consumption process.

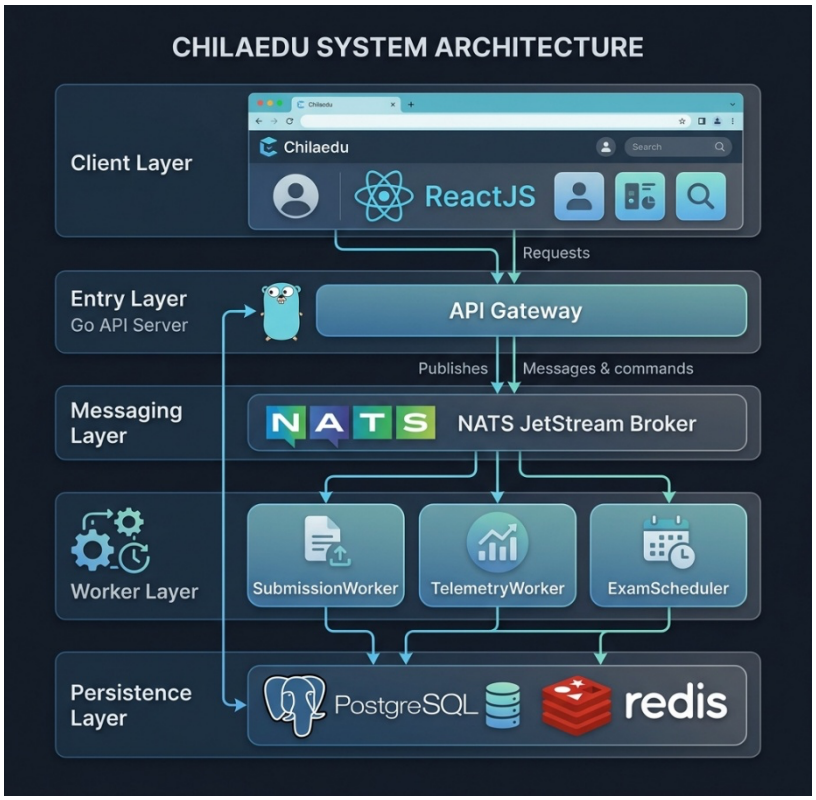

Figure 1: Global Architecture Design of the Chilaedu Platform Based on Event-Driven Architecture (EDA)

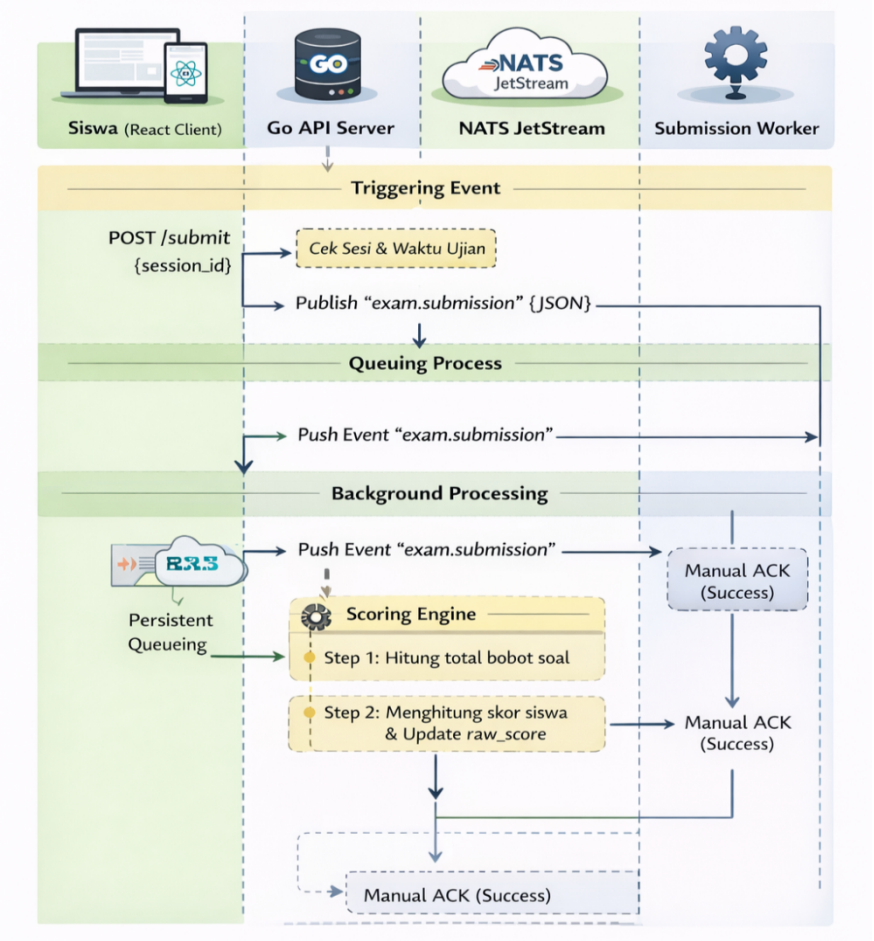

Figure 2: Asynchronous Processing Flow Diagram for Exam Answer Submission via NATS JetStream

System performance testing was conducted using a custom load-testing approach specifically designed to represent the load patterns of an online examination system. The use of this custom testing tool aimed to ensure flexibility in scenario configuration and full control over testing parameters. The testing scenario was designed to simulate request-spike conditions, with a total of 300 users sending requests simultaneously.

Variations in concurrency levels were used as an independent variable to observe system behavior under various load intensities. Testing was conducted under four concurrency-level scenarios: 10, 25, 50, and 100 simultaneous connections. Each scenario was executed using consistent parameters to ensure the validity of the result comparisons.

Table 2. Load Testing Scenario Parameters

Parameter	Value
Target Endpoint	POST /api/v1/siswa/exam-sessions/{sessionID}/submit
Total Users	300 users
Total Requests	1,200 requests
Concurrency Variations	10, 25, 50, 100
Timeout	10 seconds
Authentication	JWT

Test result data were collected and analyzed using a quantitative approach based on system performance metrics. Four main indicators were used in the evaluation: throughput, latency, error rate, and resource utilization. Throughput is defined as the number of successfully processed requests per second, reflecting the system's capacity to handle load. Latency is measured using percentile distributions (p50, p95, and p99) to provide a more representative picture of user experience. In the context of online exam systems, the p99 value is a critical indicator because it represents system performance under worst-case conditions experienced by a small subset of users.

Error rate is calculated as the percentage of failed requests out of the total requests sent, including server errors and connection failures. Meanwhile, resource utilization is measured based on CPU and memory usage during the testing process. Analysis of these metrics aims to evaluate the system's efficiency in utilizing computing resources.

Each testing scenario was executed under consistent, controlled conditions, and the resulting data were analyzed comparatively across concurrency levels to identify performance patterns under increasing load.

RESULTS AND DISCUSSION

Results

The research results describe the main findings of the study. The presentation of the results and discussion is organized systematically and includes only data and information relevant to the research objectives. The discussion section of the research article explains the results obtained from the study. A series of load tests simulating simultaneous exam answer submissions generated empirical data regarding the performance of the asynchronous architecture based on Go and NATS JetStream. The aggregated test results from all scenarios are presented in detail in Table 3.

Table 3. Tabulation of Asynchronous Event-Driven API Performance Metrics (300 Users per Scenario)

Concurrency Level	HTTP Response	Throughput (req/s)	p50 (ms)	p95 (ms)	p99 (ms)	Error Rate	CPU (%)	RAM (MB)
10	HTTP 202 Accepted	1,703.50	5.42	9.84	11.80	0.00%	24.08%	64.59
25	HTTP 202 Accepted	1,928.55	12.37	22.10	26.95	0.00%	19.25%	64.67
50	HTTP 202	**2,209.39**	21.05	30.18	34.34	0.00%	20.75%	67.74

	Accepted							
100	HTTP 202 Accepted	2,092.53	38.91	64.47	**79.02**	0.00%	18.03%	**70.05**

Discussion

The test results empirically provide a comprehensive answer to the research question regarding the effectiveness of Event-Driven Architecture (EDA) in handling high-concurrency workloads on a single-server infrastructure. The data demonstrates a substantial achievement: all 1,200 simulated exam submission requests deployed across four different scenarios (300 concurrent users each) were successfully accepted for asynchronous processing with HTTP 202 Accepted status codes. It must be emphasized that no failures were detected across all requests during testing based on HTTP response metrics at the application layer (zero error rate). This is not merely a performance statistic, but also a strong indication of system reliability under controlled test conditions.

Although this figure is derived from a simulation scenario and cannot be directly generalized to a production environment, this finding strengthens the justification for the overall system reliability within the limited experimental context. Within the framework of fault-tolerance theory, this achievement demonstrates the system's ability to maintain functionality under sustained load pressure—a resilience characteristic that is difficult to achieve with synchronous architectures under similar workloads. In monolithic architectures operating synchronously, massive traffic spikes arriving simultaneously (known as the thundering herd phenomenon) typically trigger database connection pool exhaustion or table-level deadlocks, leading to timeout responses (HTTP 503/504) and denial of service for end users.

On the other hand, the latency distribution pattern of the Chilaedu system across the three evaluation percentiles (p50, p95, and p99) demonstrates a tightly controlled and consistent performance curve. The fact that the p99 latency value remains below the latency threshold generally considered acceptable for web-based system interactions—without exhibiting uncontrolled tail-latency spikes—provides a strong indication that, even as concurrency pressure increases progressively, the Quality of Service (QoS) experienced by the least fortunate one percent of users remains within acceptable standards.

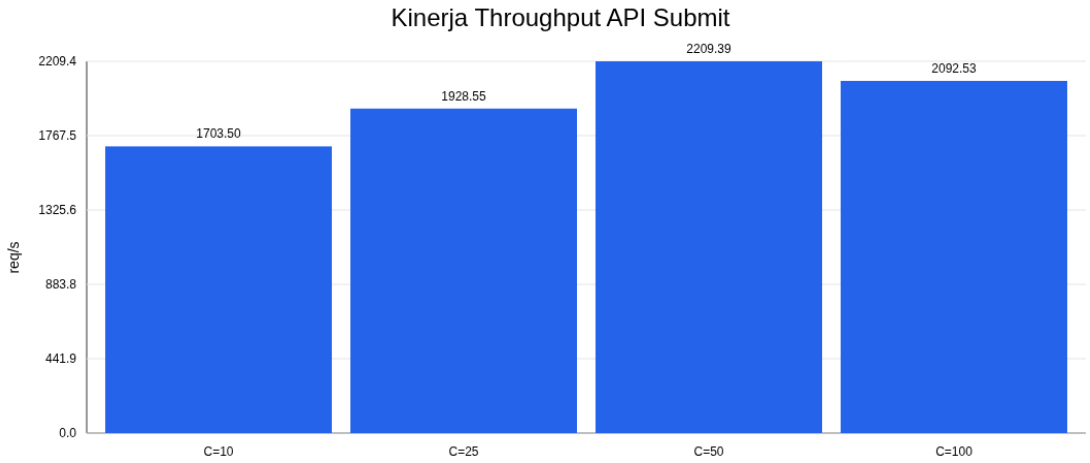

Figure 3: Throughput Curve of Chilaedu Asynchronous Integration based on NATS EDA
Source: Secondary Data, processed (2026)

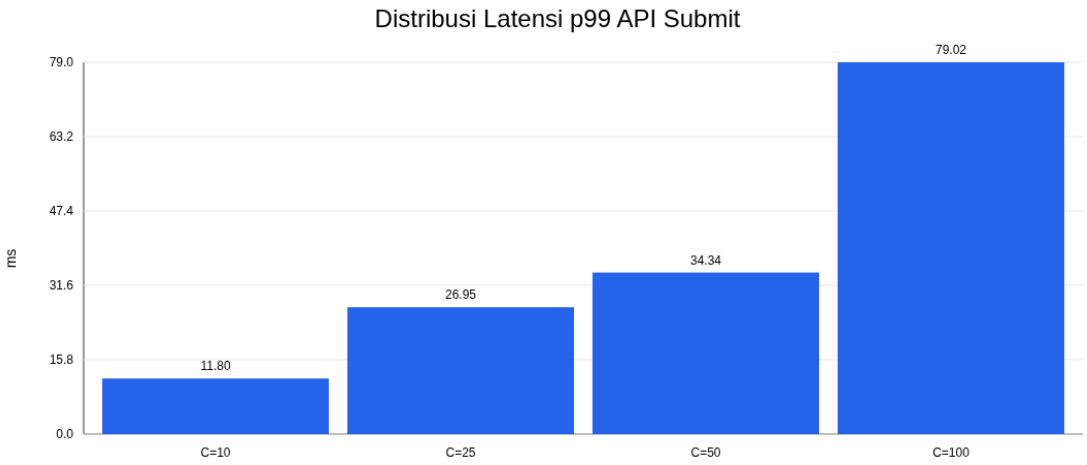

Figure 4: p99 Latency Curve of Submit API Across All Concurrency Scenarios
Source: Secondary Data, processed (2026)

Observing the scalability patterns and system behavior in greater depth, the aggregate throughput value increased proportionally from 1,703.50 req/s at concurrency level 10, peaking at its optimal point of 2,209.39 req/s at concurrency level 50. Interestingly, the system then recorded a moderate throughput decrease of 5.28%, from 2,209.39 to 2,092.53 req/s, when forced to handle an extreme load at concurrency level 100. This type of nonlinear curve behavior is consistent with the principles of M/M/c queueing theory modeling. In that model, the resource utilization rate is defined as:

$$\rho = \lambda / (c \times \mu) \quad (1)$$

where λ is the request arrival rate, μ is the service rate per computing unit, and c is the number of available CPU cores. Assuming a peak load of $\lambda \approx 2{,}209$ req/s on an architecture with $c = 4$ cores, and an estimated empirical service rate of $\mu \approx 550$ req/s per core (as a coarse-grained estimate assuming relatively even load distribution among cores, based on the approximate throughput value divided by the number of cores), the utilization value ρ approaches 1. When $\rho \geq 1$, the system enters an overutilization phase and passes its critical saturation point. At this critical phase, adding concurrent sessions directly contributes to an increase in queue waiting time due to a sharp escalation in context-switching overhead at the processor level (Henning & Hasselbring, 2024).

Although the processing rate experienced moderate degradation, the most crucial finding is that the asynchronous queue mechanism showed no indication of catastrophic failure within the tested range. In the context of this testing, no data loss was detected at the application-level observation stage, based on API response verification and internal processing log consistency, although full verification of end-to-end delivery guarantees is beyond the scope of this experiment. This indicates a significant behavioral difference compared to the general characteristics of synchronous systems, which typically experience severe failure once ρ approaches its critical threshold.

Another unique characteristic recorded was a decrease in CPU utilization, from 24.08% (at concurrency level 10) to 18.03% (at concurrency level 100), despite the objectively higher request volume. This phenomenon is thought to be related to a combination of several factors, including Go runtime scheduler efficiency, I/O wait dominance, and an increased distribution of blocked-state goroutines when the queue reaches peak capacity. Systemically, this phenomenon represents a coherent technical implication of applying measurable concurrency control principles through the implementation of Go channel semaphore gates. Grounded in Little's Law, this mechanism operates as an automatic backpressure system that regulates the average number of active workload entities in the system (L) through a strict admission-control scheme as follows:

$$L = \lambda \times W \quad (2)$$

Unlike the blocking model of synchronous architecture, this nonblocking approach confirms that the system's primary constraint does not stem from CPU-bound capacity exhaustion. Rather, within the context of this testing workload, the system exhibits the characteristics of an I/O-bound regulated system—an architectural property that is significantly more scalable. When the queue capacity reaches its limit, incoming work units are directed into a blocked state rather than consuming CPU cycles wastefully. This flow-control and self-throttling capability constitutes a crucial defense layer for preventing resource contention. Contemporary literature confirms that the absence of strict concurrency regulation often triggers severe latency storms due to race conditions, leading to fatal complications such as memory leaks when the system is pushed to its operational limits. The implementation of this control-orchestration pattern is consistent with the principles of structured concurrency and empirically validates the efficacy of Communicating Sequential Processes (CSP) theory as a foundation for asynchronous I/O-bound load control (Cox-Buday, 2017).

From the perspective of a comparative review of synchronous architectural designs and previous literature, the fundamental differentiator underlying the advantages of EDA in this research lies in the implementation of the temporal decoupling principle: the data reception phase is definitively separated from processing execution in the time dimension. In the commonly used synchronous relational architecture, the SubmitExam process is forced to maintain the client connection while waiting for the execution of layered database transactions that are vulnerable to row-level locking, with fluctuating processing times ranging between 50 and 200 milliseconds. In contrast, in the EDA implementation presented in this research, the API handler distributes the payload to NATS JetStream within the low-millisecond range and delegates status integrity to the principles of eventual consistency and asynchronous durability guaranteed by JetStream persistence mechanisms.

The score-calculation payload is then processed separately by the SubmissionWorker in an isolated process. This event-buffering mechanism completes the initial interaction within the low-millisecond range (sub-2 ms), effectively minimizing the potential for processing queue buildup on the hot path. However, this asynchronous approach introduces trade-offs in the form of increased system observability complexity for debugging processes, as well as challenges in managing temporal data consistency due to the eventual consistency mechanism.

The recorded throughput capacity of 2,209 req/s represents a potential capacity far exceeding realistic examination-load scenarios while simultaneously providing a safe overprovisioning margin. As part of capacity-planning formulation, if an institution with 500 students submits examinations within a 5-minute window, the average generated load would be only ≈ 1.67 req/s. Nevertheless, in real-world examination scenarios, submission distribution is often uneven and dominated by extreme bunching during the final seconds (burst submission). The measured capacity of 2,209 req/s indicates a sufficient buffer margin to accommodate such irregular load-distribution anomalies.

From a critical comparative perspective, this finding is not merely aligned with the literature but demonstrates substantial practical superiority. The research conducted by Cabane (2024) concluded the general superiority of event-driven architecture in distributed cluster contexts, whereas this research demonstrates that grade-A-equivalent performance can be achieved using a significantly more economical single-server topology. This distinction represents a scientific contribution that differentiates this research from previous literature.

Furthermore, in response to the structural failure caused by fatal throughput degradation in monolithic systems due to thread-pool exhaustion, as described by Blinowski (2022), this research demonstrates that an asynchronous intermediary layer such as NATS JetStream provides superior backpressure protection without requiring complex cluster orchestration. However, it should be carefully noted, with appropriate scientific humility, that this research has not explored the dimension of cross-datacenter failover recovery, which constitutes a major strength of Kubernetes-based solutions, as studied by (Arief et al., 2024).

In the realm of examination integrity, anomaly telemetry flows such as browser-tab switching were transparently distributed to the TelemetryWorker via the telemetry.event topic,

thereby providing a relevant infrastructure foundation for the development of fraud-detection systems as recommended by Ramzan (2024), without interfering with the main API.

Shifting focus to computing-resource efficiency, the data demonstrate a striking level of frugality. Main memory consumption remained stable within the range of 64.59 to 70.05 MB throughout all testing scenarios. This consistency stems from the memory efficiency under concurrency provided by the Go goroutine architecture, which employs an adaptive stack-growth scheme beginning at only 2 KB per work unit, thereby allowing memory escalation to be automatically moderated according to actual demand (Cox-Buday, 2017).

The fact that memory usage increased by only 5.46 MB (8.4%) despite a 10-fold increase in simultaneous connection density provides initial empirical evidence that space-complexity growth remains practically near constant within the conducted testing range. Although this reflects bounded growth rather than a purely asymptotic O(1) limit, its efficiency profile substantially exceeds that of conventional models.

This finding aligns with evaluative analyses of modern message brokers confirming that cloud-native technologies such as NATS JetStream are designed to minimize memory footprint compared to traditional solutions, making them highly suitable for resource-constrained distributed environments (Goel, 2024). This efficient memory-usage characteristic is a determining factor in the stability of modern distributed systems; an efficient memory footprint is essential for mitigating the risk of cascading failure originating from Out-of-Memory (OOM) crises.

This performance contrasts sharply with traditional operating-system thread-pool models, which require static memory allocation ranging from 1 to 8 MB per thread—a characteristic that drastically limits the upper bound of concurrency on hardware with limited memory capacity.

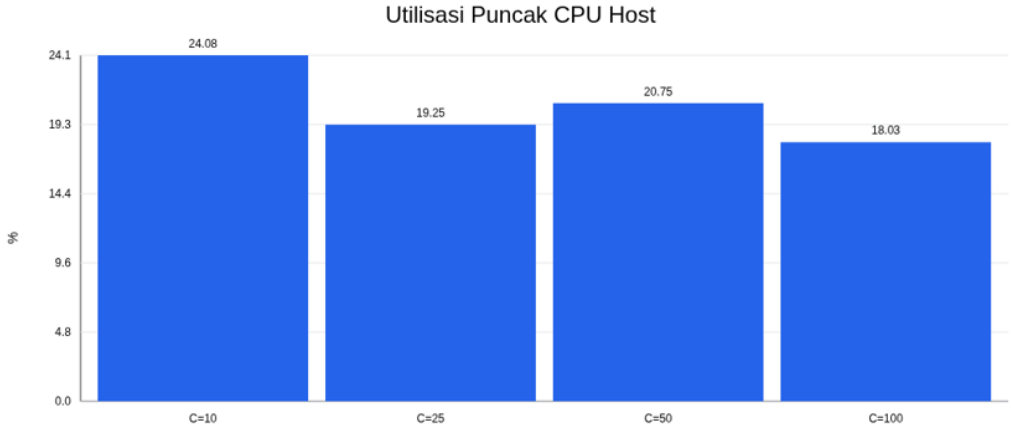

Figure 5: Peak Host CPU Utilization Profile during Load Testing
Source: Secondary Data, processed (2026)

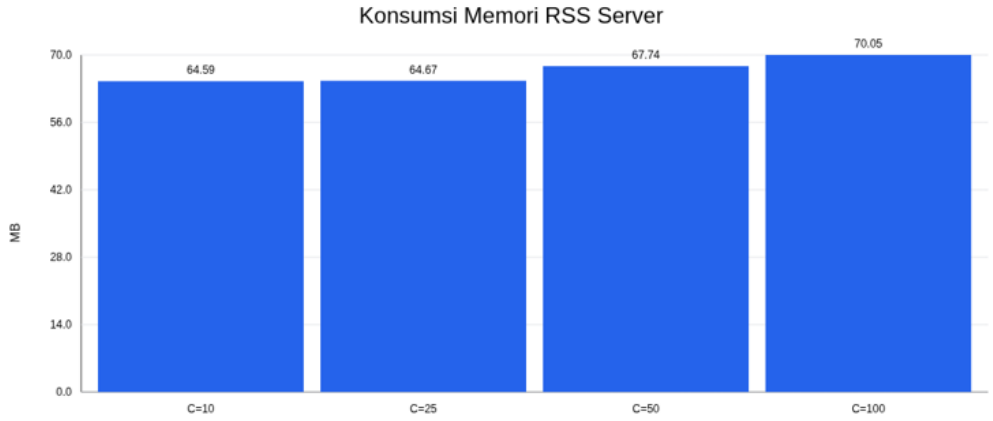

Figure 6: Peak RSS Memory Consumption Profile of the Go Server Process
Source: Secondary Data, processed (2026)

Peak CPU utilization not exceeding 24.08% underscores the availability of ample headroom capacity for subsequent organic traffic growth without the need for additional hardware. This analysis culminates in an optimal cost-efficiency ratio: the prototype architecture succeeded in delivering throughput approaching the characteristics of high-scale production systems while maintaining a resource consumption profile equivalent to that of commodity hardware.

Referring to performance-per-dollar projections, this configuration enables the operation of online exam infrastructure on affordable, mid-range Virtual Private Server (VPS) specifications, providing a tangible solution for educational institutions with limited infrastructure budgets. This achievement aligns with the evaluation criteria in the Performance and Efficiency quadrants of the PIECES Framework Asro (2024), where the proposed architecture demonstrates minimal computational power expenditure during operational spikes.

Furthermore, the available CPU headroom becomes a crucial technical prerequisite for the future integration of heavy processing modules, such as IoT-based biometric identity verification, the implications of which are mapped by Rukhiran (2023), considering that their computational load can be fully delegated to asynchronous workers without disrupting hot-path API responsiveness.

Overall, this cross-scenario analysis crystallizes into a unified insight: the combination of high throughput (2,209 req/s), stable tail-latency distribution at p99 (< 80 ms)—which, in the context of non-real-time web interaction, remains well within user tolerance ranges—and minimal CPU and RAM consumption profiles indicates that the asynchronous event-driven approach acts as a primary architectural component, rather than merely an additional optimization. This configuration closes the research gap concerning the feasibility of event-driven architecture (EDA) implementation on resource-constrained infrastructure while simultaneously providing a practical formula for building robust and economical online exam infrastructure for mid-scale educational institutions.

Internal and external validity of the experiment were established through strict variable control in a controlled environment. Regarding internal validity, the test environment was isolated on a local network to minimize external latency fluctuations (jitter); machine specifications were kept static without foreign background processes that could interrupt CPU and memory consumption; the database and NATS JetStream state were restored to a clean state before each testing scenario to prevent data accumulation from affecting query performance; and the use of automated load-testing scripts ensured that each scenario was executed using identical and repeatable procedures.

However, this analysis has not yet included measurements of statistical variability, such as confidence intervals or standard deviation; therefore, interpretation is focused on aggregate empirical trends. Regarding external validity, the observed throughput behavior pattern can be specifically extrapolated to the domain of I/O-bound systems, systems with event-driven workloads, and systems vulnerable to burst-traffic scenarios. This specific profile could potentially be replicated reasonably in other public-service domains with similar challenges, such as e-government systems experiencing seasonal transaction spikes.

The limitations of this research need to be considered as part of scientific caution. First, the use of synthetic user-traffic simulation methods potentially carries modeling bias, as the generated load profile is constant and uniform, thereby not representing the complexity of real user behavior, such as random navigation, repeated access, or double-click synchronization anomalies that could generate different I/O load patterns. Second, testing in a single-node server environment does not encompass the dynamics of inter-node consensus overhead, which is a typical challenge in large-scale distributed clusters. Lastly, the removal of public-network latency variables (packet loss and geographical jitter) from the test environment, while intended to isolate internal architectural performance, means that in real-world ecosystem implementation, end-user connection stability remains a significant external variable beyond the reach of software-layer optimization.

The theoretical implications of this study provide crucial empirical evidence regarding system scalability in constrained environments, which in turn challenges the perception that high

scalability always requires expensive physical infrastructure expansion. This finding reinforces a paradigm shift toward the concept of software-defined scalability: careful architecture-level optimization design—including the implementation of temporal decoupling, near-constant-growth memory management, and strict concurrency control—has leverage as significant as that of brute-force infrastructure-scaling solutions. In some dimensions, this approach even surpasses the effectiveness of pure hardware-capacity expansion. Furthermore, this research provides an empirical model for the application of backpressure and flow-control strategies at the application level, strengthening the theory of Communicating Sequential Processes (CSP) in the real-world context of the education sector and proving that functional decoupling through event-driven architecture (EDA) can increase system resilience against the thundering herd phenomenon.

Future research directions can be directed along three more specific main paths. First, exploration of event-rate-driven autoscaling mechanisms that dynamically adjust computing capacity based on the real-time rhythm of telemetry arrival rates (λ), rather than relying on reactive and lagging CPU utilization thresholds. Second, implementation of adaptive concurrency-control algorithms with variable-state throttling capable of dynamically adjusting semaphore limits based on current system conditions, as well as integration of latency-aware QoS scheduling to prioritize processing based on response-time urgency. Third, embedding artificial intelligence for predictive failure-monitoring functions leveraging the telemetry data stream from NATS JetStream to shift the system's posture from reactive self-healing toward proactive self-optimizing provisioning in line with the dynamics of actual field-usage patterns.

CONCLUSION

Load test results showed that the proposed architecture was able to maintain a throughput of 2,209 requests per second with a 0.00% HTTP failure rate under peak load conditions of 100 simultaneous connections. Latency performance at the 99th percentile (p99) was recorded at 79.02 ms, which is below the 100 ms threshold, thereby preventing user interface disruptions (browser freezes). The efficiency of computing resource usage is reflected in the peak CPU utilization of 18.03% and maximum memory consumption of 70.05 MB, both of which remained within safe operational limits for server infrastructure. This research implemented and empirically validated an Event-Driven Architecture (EDA) using Go and NATS JetStream to address the synchronized spike problem in online exam answer collection. Testing across 300 synthetic users confirmed a peak throughput of 2,209 req/s, p99 latency of 79.02 ms (below the 100 ms threshold), a 0.00% HTTP error rate, peak CPU utilization of 18.03%, and maximum RAM usage of 70.05 MB—all within safe operational bounds on a single-server prototype. The principal contribution of this research is the empirical demonstration that a NATS JetStream-based EDA deployed on constrained single-server infrastructure can deliver competitive reliability without requiring complex orchestration, thereby constituting a viable and affordable solution for mid-scale educational institutions.

The main contribution of this research is the empirical demonstration that implementing NATS JetStream as an asynchronous intermediary layer on a limited-capacity single server can achieve competitive performance without requiring complex cluster infrastructure. This approach makes NATS-based EDA a viable and affordable alternative for mid-scale educational institutions. Furthermore, the smooth operation of online exam infrastructure represents a tangible manifestation of improved public service efficiency, which has implications for reducing student dissatisfaction rates and consequently contributes to enhancing the performance and institutional loyalty of educational institutions at a macro level (Rukhviyanti et al., 2022; Rukhviyanti, 2025).

This research has several limitations that should be considered when interpreting its results. First, all testing was conducted in an isolated virtual environment (Microsoft Hyper-V), so the results may differ under production conditions involving heterogeneous data traffic. Second, the testing scenario simulated only the submit operation without involving other simultaneous user interactions. Third, this research did not include recovery testing or fault-tolerance scenarios for the NATS JetStream component. For future research, this asynchronous architectural foundation could be integrated with machine learning classification algorithms to develop a background queue-based automated proctoring system capable of proactively detecting fraud

anomalies in real time.

ACKNOWLEDGEMENT

The authors express sincere gratitude to SMK Kiansantang Bandung for providing the institutional case study context and access to operational requirements data. The authors also thank the Chilaedu platform development team for technical support during prototype implementation, and all reviewers for constructive feedback that substantially improved this manuscript. No external funding was received for this research.

AUTHOR CONTRIBUTION STATEMENT

If any, Author Author 1: Conceptualisation, system architecture design, software engineering, load-testing implementation, data analysis, and original manuscript drafting. Author 2: Supervision, theoretical framework development, results validation, manuscript review and revision, and final approval. Both authors have read and agreed to the final version of this manuscript.

REFERENCES

Abdelsalam, M., Shokry, M., & Idrees, A. M. (2023). A proposed model for improving the reliability of online exam results using blockchain. *IEEE Access*, *12*, 7719–7733. https://doi.org/10.1109/ACCESS.2023.3304995

Arief, S. N., Prasetyo, A., Mashudi, I. A., & Nugraha, B. S. D. (2024). Analisa Performansi Ujian Online Berbasis Website Dengan Arsitektur Lightweight Kubernetes (K3S). *Jurnal Minfo Polgan*, *13*(2), 1739–1745. https://doi.org/10.33395/jmp.v13i2.14240

Asro, A., Monica, M., Rukhviyanti, N., & Yusron, M. (2024). Analisis Literatur Review Perencanaan Strategi Sistem Informasi Menggunakan Metode Pieces Framework. *KRESNA: Jurnal Riset Dan Pengabdian Masyarakat*, *4*(2), 161–169. https://doi.org/10.36080/kresna.v4i2.182

Blinowski, G., Ojdowska, A., & Przybyłek, A. (2022). Monolithic vs. microservice architecture: A performance and scalability evaluation. *IEEE Access*, *10*, 20357–20374. https://doi.org/10.1109/ACCESS.2022.3152803

Cabane, H., & Farias, K. (2024). On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems*, *153*, 52–69. https://doi.org/10.1016/j.future.2023.10.021

Cox-Buday, K. (2017). *Concurrency in Go: Tools and Techniques for Developers*. " O'Reilly Media, Inc."

Elumalai, K. V., Sankar, J. P., Kalaichelvi, R., John, J. A., Menon, N., Alqahtani, M. S. M., & Abumelha, M. A. (2021). Factors affecting the quality of e-learning during the COVID-19 pandemic from the perspective of higher education students. *COVID-19 and Education: Learning and Teaching in a Pandemic-Constrained Environment*, *189*(3), 169.

Goel, R. (2024). Evaluating Message Brokers: Performance, Scalability, and Suitability for Distributed Applications. *American Journal of Computer Architecture*, *11*(5), 62–65. https://doi.org/10.5923/j.ajca.20241105.02

Hasbi, H., Rukhvianti, N., & Gunawan, H. (2020). Pembinaan Motivasi Belajar Siswa Menggunakan Metode ARCS. *E-Dimas: Jurnal Pengabdian Kepada Masyarakat*, *11*(3), 254–259. https://doi.org/10.26877/e-dimas.v11i3.5653

Henning, S., & Hasselbring, W. (2024). Benchmarking scalability of stream processing frameworks deployed as microservices in the cloud. *Journal of Systems and Software*, *208*, 111879. https://doi.org/10.1016/j.jss.2023.111879

Kleppmann, M. (2017). *Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems*. " O'Reilly Media, Inc."

Lai, C. F., & Huynh, P. V. (2022). *Air Traffic Management TestBed: Messaging Performance*. NASA.

Njuguna, A. M. (2022). User experience of online examinations and proctoring: A case based study. *International Journal of Current Science Research and Review*, *5*(7), 2326–2335.

Oliveira Rocha, H. F. (2021). Choosing the Correct Event Schema Design in Event-Driven Microservices. In *Practical Event-Driven Microservices Architecture: Building Sustainable and Highly Scalable Event-Driven Microservices* (pp. 323–355). Springer.

https://doi.org/10.1007/978-1-4842-7468-2_8
Oktaviane, S. P., & Rukhviyanti, N. (2026). The Role of Information Systems in Managing Student Report Card and Attendance Data in Junior High Schools. *Jurnal Ilmiah Global Education*, *7*(1), 402-413. https://doi.org/10.55681/jige.v7i1.5211
Paredes, A. J., Vargas Escobar, L. A., Inciarte González, A., & Mercado Porras, C. (2023). Assessment of learning in online academic programs from the digital transformation impelled by Covid-19. *Revista de Ciencias Sociales*, *29*(1), 18–34.
Ramzan, M., Abid, A., Bilal, M., Aamir, K. M., Memon, S. A., & Chung, T.-S. (2024). Effectiveness of pre-trained CNN networks for detecting abnormal activities in online exams. *IEEE Access*, *12*, 21503–21519. https://doi.org/10.1109/ACCESS.2024.3359689
Rukhiran, M., Wong-In, S., & Netinant, P. (2023). IoT-based biometric recognition systems in education for identity verification services: Quality assessment approach. *Ieee Access*, *11*, 22767–22787. https://doi.org/10.1109/ACCESS.2023.3253024
Rukhviyanti, N. (2025). Balanced scorecard: Performance measurement of university mediated by student loyalty. *Jurnal Manajemen*, *29*(2), 400-420. https://doi.org/10.24912/jm.v29i2.2694
Ruhviyanti, N., Wasliman, I., Hanafiah, H., & Tejawiani, I. (2022). Implementation of the balanced scorecard in improving the performance of private universities. *International Journal of Educational Research & Social Sciences*, *3*(3), 1242-1246. https://doi.org/10.51601/ijersc.v3i3.390
Sahni, S., Verma, S., & Kaurav, R. P. S. (2025). Understanding digital transformation challenges for online learning and teaching in higher education institutions: a review and research framework. *Benchmarking: An International Journal*, *32*(5), 1487–1521. https://doi.org/10.1108/BIJ-04-2022-0245
Shard, Kumar, D., & Koul, S. (2024). Digital transformation in higher education: A comprehensive review of e-learning adoption. *Human Systems Management*, *43*(4), 433–454. https://doi.org/10.3233/HSM-230190
Vom Brocke, J., Hevner, A., & Maedche, A. (2020). *Design science research. Cases*. Springer. https://doi.org/10.1007/978-3-030-46781-4