Paradigma, Vol. 28, No. 1, March 2026, P-ISSN 1410-5063, E-ISSN: 2579-3500, Page. 13-21
Published: LPPM Universitas Bina Sarana Informatika

PARADIGMA

website: http://jurnal.bsi.ac.id/index.php/paradigma/

Interactive Live Streaming Website with Cloud Computing and AI Assistant Integration

Satria Tantra Adlillah[1], Ai Rosita[1]

[1]Faculty of Informatics Engineering, Universitas Widyatama, Bandung, Indonesia

ARTICLE INFORMATION

Artikel History:
Received: 14-08-2025
Revised: 17-09-2025
Accepted: 05-03-2026
Available Online: 31-03-2026

Keyword:

Live Streaming
Interactive Website
Cloud Computing
AI Assistant
Real-time Interaction

ABSTRACT

Live streaming platforms continue to face persistent challenges related to high latency, limited scalability, and insufficient real-time interactivity, which degrade user experience and hinder audience engagement. This study aims to address these limitations by developing StreamCats, a third-party interactive web-based live streaming system that integrates cloud computing infrastructure, WebSocket-based real-time communication, and an AI assistant. The system was developed using Object-Oriented Methodology (OOM) and the Rational Unified Process (RUP), implemented with Next.js, Socket.IO, Supabase, and Puter.js, while TikTok live event connectivity was achieved through WebSocket reverse engineering. System performance was evaluated through stress testing using Apache JMeter under 10, 50, and 100 concurrent user scenarios, functional validation of six core modules, and usability testing involving ten participants. The results demonstrate that under 100 concurrent users, the system maintained an average latency of 2,180.3 ms (well below the 5,000 ms threshold), a response time of 139 ms, CPU usage of 1.13%, and a throughput of 89.8 requests per second, outperforming comparable platforms Indofinity (57 req/s) and Tikfinity (19.5 req/s). All six modules passed functional tests, and 90% of participants successfully completed usability tasks with an average comprehension time of under five minutes. These findings confirm that the integration of cloud computing and AI-driven assistance significantly enhances scalability, responsiveness, and user experience in interactive live streaming environments, establishing StreamCats as a viable and competitive foundation for multi-platform real-time streaming solutions.

Corresponding Author:

Satria Tantra Adlillah,
Informatics Engineering,
Universitas Widyatama,
Jl. Cikutra, No. 204A, Sukapada, Cibeunying Kidul, Bandung, Indonesia, 40125,
Email: satriatantra02@gmail.com

INTRODUCTION

The rapid advancement of digital technology has significantly transformed the ways in which people communicate, create, and interact. One of the most notable manifestations of this transformation is the rise in popularity of live streaming services, which have expanded far beyond their initial entertainment-focused scope into education, digital marketing, and community-based interactions (Mardiah et al., 2023). In this context, Massive amounts of data are used for network-based video/live streaming. Most importantly, uninterrupted streaming feeds and data must use low latency channels to prevent end users' video playback from buffering(Kumar, 2024). Platforms such as TikTok, YouTube Live, and Instagram Live have become primary mediums for content creators to share ideas, showcase creative works, and build direct relationships with audiences. This shift in user behavior indicates that viewers no longer wish to remain passive consumers but increasingly seek active

DOI: https://doi.org/10.31294/p.v28i1.9724

participation in live broadcasts. In this context, one key characteristic of live-streaming is that it enables a streamer to interact with a massive audience group in real-time (Wang et al., 2021).

Despite these developments, live streaming platforms still face persistent challenges. High latency, limited scalability, and insufficient support for interactive features remain major obstacles, while the computational burden on client devices often restricts user accessibility. These problems reduce the overall quality of experience, hinder audience engagement, and limit the effectiveness of live streaming as a medium for communication and commerce. Previous studies have highlighted the importance of interactivity in shaping consumer trust and purchase intention, demonstrating that consumer–streamer and consumer–consumer interactions play a significant role in engagement during live streaming sessions (Liu & Zhang, 2024). as well as the role of anthropomorphic virtual influencers in enhancing immersion during live streaming sessions (Xie & Desouza, 2025). However, most of these studies emphasize behavioral or marketing outcomes rather than offering technical frameworks capable of directly solving latency, scalability, and interactivity issues.

Cloud computing emerges as a promising solution to address these technical challenges. By centralizing data processing, cloud infrastructures can reduce client-side loads, improve scalability, and support adaptive resource allocation (Baccour et al., 2021; Huang et al., 2022). In parallel, Artificial Intelligence (AI) assistants provide additional support by enabling contextual recommendations, adaptive engagement strategies, and real-time interaction management(Wang et al., 2021; Farahani et al., 2024). Together, these technologies have the potential to create third-party live streaming platforms that not only enhance performance but also deliver inclusive, user-friendly experiences for both novice and professional streamers.

Given these gaps, the objective of this study is to develop *StreamCats*, an interactive web-based live streaming system that integrates cloud computing, WebSocket-based real-time data communication, and a JavaScript-powered AI assistant. The system is designed to achieve low-latency interaction, scalable performance across multiple platforms, and adaptive user support. By combining cloud resource optimization and AI-driven engagement, the research contributes to bridging the gap between theoretical insights on interactivity and practical technical solutions for real-time live streaming environments.

RESEARCH METHOD

Definition and Importance of Research Method

Research methods refer to systematic approaches employed to collect, analyze, and interpret data to answer research questions or validate hypotheses (Irawan et al., 2024). A well-defined research methodology ensures that results are reliable, contribute to existing knowledge, and provide actionable solutions (Mishra & Mishra, 2024). In this study, the research method integrates Object-Oriented Methodology (OOM), Rational Unified Process (RUP), reverse engineering, and web scraping to develop and evaluate a Website called StreamCats that have interactive live streaming system. This combined approach ensures modular, structured design, real-time performance, and accurate simulation of user interactions. The research method carried out is shown in Figure 1.

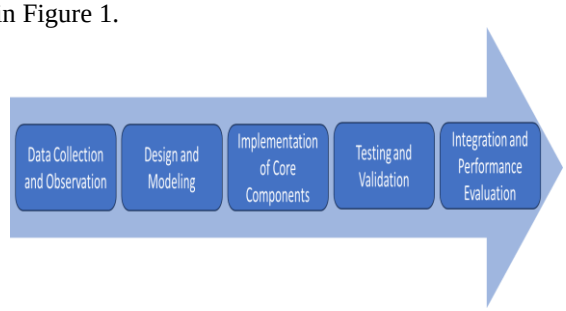

Figure 1 Research Stages

The research was conducted in five structured stages, beginning with data collection and observation, followed by system design and modeling, implementation of core components, testing and validation, and finally integration and performance evaluation. Each stage is flexible and adaptable based on iterative findings, allowing the research process to refine and optimize the development of the StreamCats interactive live streaming system.

1. Data Collection and Observation

Data collection commenced with direct observation of TikTok live streaming sessions to understand event patterns, user interactions, and data transmission between clients and servers. Reverse engineering techniques were applied using network inspectors and packet sniffers to capture communication payloads and identify event structures, including gifts, comments, and likes (Yu et al., 2023). This stage provides a foundation for system design and ensures that the implemented event handlers reflect real user behaviors.

2. Design and Modeling

The system architecture was modeled using Object-Oriented Methodology (OOM) to ensure modularity, encapsulation, and reusability of components (Al-Msie'deen, 2018; Husna et al., 2024). The Rational Unified Process (RUP) guided the iterative and incremental development across four phases: Inception, Elaboration, Construction, and Transition (Prihartono & Utami, 2023; Karunamoorthy & Prakash, 2021). This methodology enables early validation of requirements, continuous refinement of architecture, and effective risk management. The mapping of RUP phases to research activities is shown in Figure 2

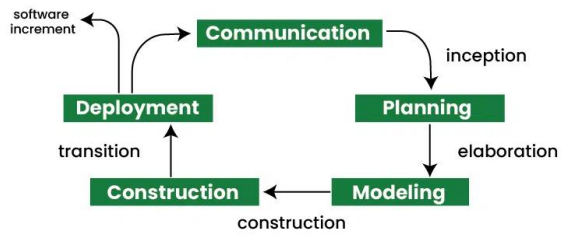

Figure 2 Rational Unified Process

At Figure 2, Each phase has specific objectives, deliverables, and activities, ensuring that the system is developed in a controlled, systematic manner.

a. Inception Phase
   The inception phase focuses on understanding the project scope, objectives, and feasibility. Activities include gathering initial requirements, defining the project vision, and identifying risks. For the StreamCats platform, this phase ensures clarity on the multi-platform live streaming goals, AI assistant features, and integration with TikTok through reverse engineering. Deliverables typically include a project vision document, initial use cases, and feasibility analysis.
b. Elaboration Phase
   In the elaboration phase, the system architecture is designed and validated. Critical requirements are refined, risks are addressed, and the project plan is detailed. In the context of StreamCats, this phase involves modeling the architecture using Object-Oriented Methodology (OOM), designing WebSocket communication modules, and defining AI assistant workflows. Key deliverables include architectural models, refined use cases, and a risk mitigation plan.
c. Construction Phase
   The construction phase focuses on building and implementing the system components. Incremental development allows modules to be implemented, integrated, and tested iteratively. For StreamCats, this includes coding the WebSocket handlers, AI assistant module, and web scraping functionality. The phase emphasizes modularity, maintainability, and performance optimization. Deliverables include functional software components, tested modules, and updated documentation.
d. Transition Phase
   The transition phase ensures that the system is deployed, tested in real-world conditions, and accepted by users. Activities involve system integration, user training, and final performance testing. In StreamCats, this phase validates that the AI assistant responds correctly, real-time events are handled efficiently, and Supabase database operations are consistent and secure. Deliverables include deployed system, user manuals, and final evaluation reports.

The selection of Object-Oriented Methodology (OOM) and Rational Unified Process (RUP) in this research is based on their suitability for addressing the problems of latency, scalability, and interactivity in live streaming systems. OOM ensures modularity, encapsulation, and reusability, allowing independent components such as the WebSocket event handler, AI assistant, and database integration to be developed and optimized without affecting the entire system(Al-Msie'deen, 2018). Meanwhile, RUP provides an iterative and risk-driven development cycle, enabling continuous validation of requirements such as low latency and high scalability, while still maintaining structured documentation (Prihartono & Utami, 2023). Compared to Waterfall, which is too rigid, or Agile, which often lacks formal documentation, RUP offers a balanced approach that is particularly effective for projects requiring both flexibility and clear deliverables. This combination provides the methodological foundation necessary for developing StreamCats as a third-party platform that enhances interactivity in live streaming.

3. Implementation of Core Components

Key components implemented include:

a. WebSocket Event Handler-manages real-time events under high-load conditions with minimal latency.
b. AI Assistant Module-developed using AI-as-a-Service (AIaaS) with Natural Language Processing (NLP) to provide context-aware assistance for streamers. Session data is managed externally via Supabase.
c. Web Scraping and Simulation – Optimized web scraping methods are employed to collect interaction patterns and metadata for testing, following state-of-the-art strategies in real-time data extraction from dynamic web platforms.

This approach allows realistic simulation of live streaming scenarios for subsequent performance evaluation. For example, AI-driven virtual anchors in live-streaming e-commerce environments have been shown to enhance consumer engagement and increase perceived playfulness, leading to higher levels of interaction and satisfaction (Xie & Desouza, 2025). To support similar levels of interactivity in this research, it was necessary to ensure that event data from platforms such as TikTok could be captured and replicated with high fidelity. Reverse engineering was therefore adopted to understand and replicate the data flow of TikTok live streaming events, including comments, gifts, and likes. Since official APIs do not provide full access to real-time interaction data, reverse engineering using network inspectors and packet sniffers offers a practical way to capture and interpret event payloads directly (Yu et al., 2023; Siala et al., 2024). This ensures that the system reflects actual user behavior and addresses the core problem of interactivity with minimal latency.

In addition, WebSocket technology was integrated as the core connector between TikTok events and the StreamCats platform. Unlike traditional HTTP polling, WebSocket enables persistent,

bidirectional communication, allowing real-time delivery of comments, likes, and virtual gifts with minimal delay. This low-latency communication channel is crucial for preserving the immediacy of live interactions, a factor repeatedly identified in prior research as essential to maintaining user engagement in streaming environments (Farahani et al., 2024; Maulidina & Nugraha, 2023).

Artificial Intelligence (AI) assistance was also incorporated to improve usability for streamers who may not possess advanced technical skills. By leveraging Puter.js for natural language understanding, the system provides contextual guidance, automates repetitive tasks, and simplifies the configuration of interactive events. Prior studies have shown that AI assistants can reduce cognitive load, enhance decision-making, and increase participation in live-streaming contexts (Wang, He, et al., 2021; Wang, Huang, et al., 2023; Mei et al., 2025). For StreamCats, this ensures that the platform remains inclusive and accessible to both novice and professional users, aligning with the project's objective of lowering technical barriers to interactive live streaming.

Web scraping was also employed to simulate realistic usage scenarios and to collect dynamic interaction patterns from live streaming environments. Unlike static datasets, scraped data captures live variations in event frequency and user engagement, which are critical for testing scalability and responsiveness. Recent studies highlight that optimized scraping techniques can support real-time analysis and system testing in dynamic platforms (Xie & Desouza, 2025; Khder, 2021). By combining reverse engineering, WebSocket integration, AI assistance, and web scraping, this research ensures both the accuracy of event handling and the realism of performance evaluation.

4. Testing and Validation (Procedural Focus)

Controlled experiments simulated multiple concurrent clients to assess the system's performance under load. Key metrics included latency, throughput, memory usage, and response time, while usability tests evaluated interface responsiveness, navigation, and user experience across different devices. Functional testing ensured that the AI assistant, WebSocket handlers, and web scraping modules operated according to design specifications (Wijaya, 2021; Fedianto et al., 2023; Hermawan & Alim, 2021). The technical metrics can be seen on Table 1

Table 1 Technical Metrics

Name	Metric Type	Target	Description	Measurement Method
M1	Latency (ms)	≤ 5000 milliseconds (5 seconds)	Delay from TikTok event received to UI response	Timestamp comparison (WebSocket)
M2	Response Time (ms)	≤ 500 milliseconds	User action to system visual response	Timestamp comparison
M3	CPU Usage (%)	≤ 30% under the scenario of 50 simultaneous users	Processor load during interaction	Task Manager, htop, Supabase
M4	Bandwidth / Throughput (bps)	< 500 kilobits per second (kbps) per user	Data transfer rate during streaming	Apache JMeter packet analyzer
M5	User Load Simulation	10, 50, and 100 simultaneous users	Scalability & stability under varying load	Apache JMeter traffic simulation

Table 1 explains the system performance evaluation metrics used in this study. The first metric, latency, is targeted at no more than 5000 milliseconds, measured as the time difference between the reception of an event from TikTok via WebSocket and the corresponding system response, such as executing an action or visualizing a notification on the user interface. The second metric, response time, is set with a stricter target of no more than 500 milliseconds, which refers to the duration from when a user performs an action, for example pressing a configuration button, until the system's response becomes visually apparent on the screen. The third metric concerns CPU usage, where the acceptable threshold is a maximum of 30% under the scenario of 50 simultaneous users, representing the processor resource consumption during interaction sessions. In addition, bandwidth or throughput is also evaluated, with a target of less than 500 kilobits per second per user, calculated as the amount of data transferred from the server to the client per second during live streaming to reflect the efficiency of the communication channel. Finally, user load simulation is carried out at levels of 10, 50, and 100 concurrent users to assess system scalability and stability under varying load conditions. The measurements are performed using timestamp logging for latency and response time, Task Manager and monitoring tools for CPU usage, Apache JMeter for bandwidth and throughput, as well as traffic simulation for user load testing.

5. Integration and Performance Evaluation (Procedural Focus)

After individual component validation, the StreamCats platform was integrated and evaluated holistically to ensure seamless interoperability among all modules. The evaluation procedures included verifying AI assistant responses for contextual accuracy and relevance, monitoring real-time event handlers to confirm consistency and low-latency performance, and validating Supabase database storage to guarantee both data integrity and security under concurrent access. In addition, cross-platform testing

was conducted on desktop and mobile environments to assess system responsiveness, while stress tests were carried out to evaluate stability during peak interaction loads. These comprehensive procedures strengthen the confidence that the system is technically prepared for real-world deployment as a multi-platform interactive live streaming platform. Figure 3 illustrates the final integrated architecture and its corresponding data flow.

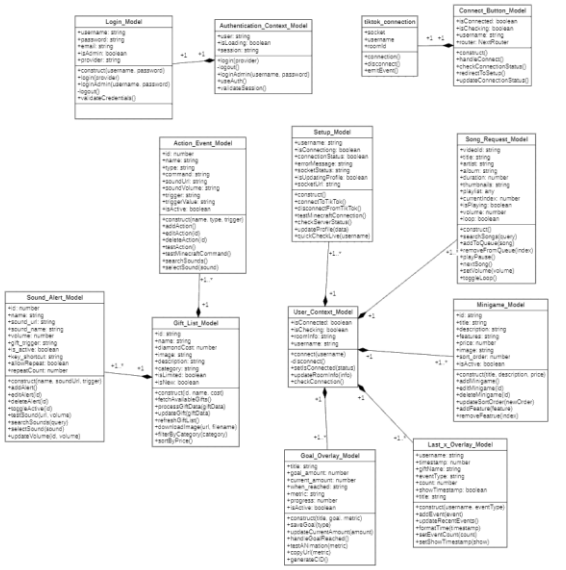

Figure 3 Class Diagrams of StreamCats

The StreamCats system architecture is designed to be modular and scalable with a client-server approach, as illustrated in the class diagram shown in Figure 3, separating responsibilities between the user interface and business logic. The frontend uses Next.js 15 and React 19 for fast rendering and component-based UI management, while the backend employs Node.js and Express.js to efficiently handle asynchronous I/O. Real-time communication is enabled through Socket.IO with low latency, which is critical for live streaming (Alharbi et al., 2021).

User data, action configurations, and interaction logs are stored in Supabase, a cloud-based PostgreSQL database that supports real-time connectivity, JWT authentication, and Row-Level Security (RLS) (Bokhari & Herraiz, 2024). External integrations include TikTok event connections via WebSocket reverse engineering, Minecraft control through the RCON protocol, and MyInstant API for audio notifications.

Cloud and AI integration leverage Supabase and Puter.js, where user inputs are processed by an external AI service and the results are returned to the UI or stored in the database. This approach supports serverless deployment on Vercel, continuous development, scalability, and resource efficiency without overloading the client.

RESULTS AND DISCUSSION

The testing of the StreamCats platform demonstrates that the system successfully meets the technical specifications set for a web-based interactive live streaming environment. The system was evaluated based on the technical performance metrics defined in Table 1, covering latency (M1), response time (M2), CPU usage (M3), bandwidth efficiency (M4), and scalability (M5). These criteria form the baseline for assessing StreamCats performance under different user loads. Stress tests were conducted using Apache JMeter to simulate concurrent user loads of 10, 50, and 100 clients. All of the simulation can be seen on Table 2.

Table 2 StreamCats Simulate 10, 50, 100 Users

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Avg. Bytes
Stream Cats	10	141	125	209	23.40	0.00 %	9.7/sec	237.69	25118.0
TOTAL	10	141	125	209	23.40	0.00 %	9.7/sec	237.69	25118.0
Stream Cats	50	151	127	248	20.25	0.00 %	44.6/sec	1094.08	25118.0
TOTAL	50	151	127	248	20.25	0.00 %	44.6/sec	1094.08	25118.0
Stream Cats	100	131	117	163	9.21	0.00 %	88.9/sec	2180.38	25118.0
TOTAL	100	131	117	163	9.21	0.00 %	88.9/sec	2180.38	25118.0

Across all scenarios, the system maintained stable performance, with CPU and bandwidth usage remaining within optimal thresholds and response times consistently within acceptable limits for real-time interaction. After a stress test simulation, the test results can be seen in Table 3

Table 2 StreamCats Stress Test

Total Users	M1	M2	M3	M4	Status
10 Users	237.69	39	0.3	9.7	Passed
50 Users	1094.08	78	0.6	44.6	Passed
100 Users	2180.3	139	1.13	88.9	Passed

The testing summarized in Table 3 was conducted based on the evaluation metrics defined in Table 1. For latency (M1), the system consistently remained below the maximum threshold of 5000 milliseconds, with an average of 237.69 ms under 10 users, 1094.08 ms under 50 users, and 2180.3 ms under 100 users, thereby meeting the performance target. Similarly, response time (M2) stayed well below the 500 ms limit, averaging 39 ms, 78 ms, and 139 ms across the respective load scenarios. CPU usage (M3) was stable at 0.3%, 0.6%, and 1.13%, far under the 30% threshold, demonstrating efficient resource management. Bandwidth and throughput (M4) also performed efficiently, remaining below the ceiling of 500 kbps per user at all test levels (9.7, 44.6, and 88.9 kbps, respectively). Finally, in the user load simulation (M5), the system operated without crashes or functional failures even at 100 concurrent users, confirming its scalability and stability. Overall, these results indicate that StreamCats remained within acceptable operational limits, particularly in CPU and bandwidth efficiency, and is capable of supporting real-time interactive streaming under varying load conditions.

To evaluate whether StreamCats outperforms

other comparable platforms, an additional benchmark was conducted under 100 concurrent user simulations, as illustrated in Figures 4, 5, and 6.

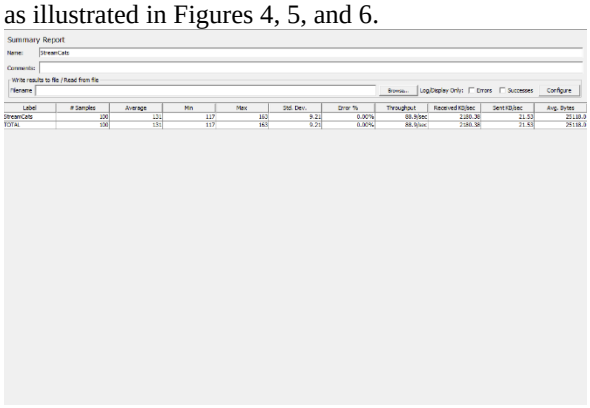

Figure 4 StreamCats 100 User Simulation Test

Figure 4 shows StreamCats performance under 100 simultaneous users. The system achieved 88.9 requests per second, the highest among the compared platforms. This demonstrates that the modular WebSocket handlers, Node.js backend, and Supabase cloud storage efficiently handle high loads, confirming StreamCats scalability, responsiveness, and readiness for intensive real-time interactive scenarios.

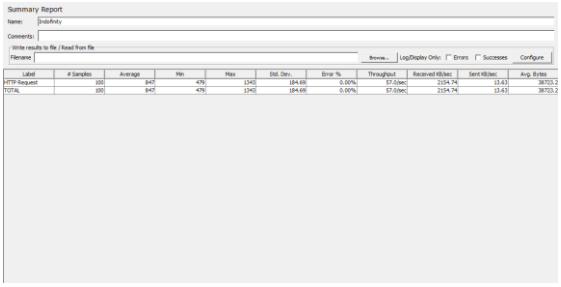

Figure 5 Indofinity 100 User Simulation Test

Figure 5 shows the stress test results for Indofinity under 100 simultaneous users using Apache JMeter. The system achieved an average throughput of 57 requests per second, indicating stable performance under high parallel loads and efficient event streaming via WebSocket.

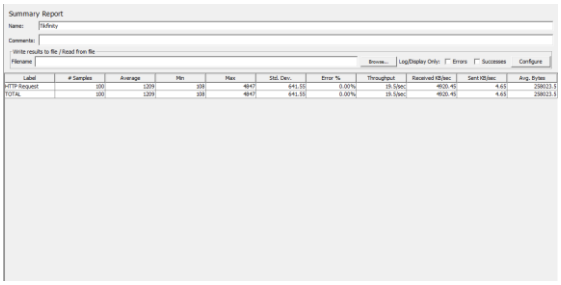

Figure 6 Tikfinity 100 User Simulation Test

Figure 6 shows Tikfinity's throughput performance with the same 100-user scenario. The platform reached only 19.5 requests per second, significantly lower than StreamCats, reflecting limited capacity for handling concurrent connections and real-time event processing, which may affect interactive live streaming experiences.

The benchmark results show that StreamCats consistently outperforms both Indofinity and Tikfinity in handling high user loads. This performance advantage can be attributed to its modular WebSocket event handler, which minimizes blocking operations and enables efficient distribution of requests across clients. Such an architecture enhances throughput while maintaining low latency, aligning with the findings of Farahani et al. (2024) that hybrid WebSocket–cloud designs provide superior responsiveness compared to traditional centralized approaches. Consequently, StreamCats demonstrates higher data transfer efficiency and stronger load-handling capacity, supported by its optimized communication pipeline and scalable backend design.

Functional testing of the platform confirms that all critical modules operate as intended, ensuring that the system meets its specified requirements and performance benchmarks. The WebSocket event handler consistently processes real-time events with minimal latency, allowing for smooth and uninterrupted data transmission between the client and server. The AI assistant, integrated through Puter.js, demonstrates reliable comprehension and contextual response generation, accurately addressing user prompts in both simple and complex interaction scenarios. Sound alerts, implemented via the MyInstant service, trigger promptly and without audio distortion, while RCON commands execute correctly within the Minecraft integration, maintaining stable server communication. Supabase storage remains secure through enforced access control policies and consistent through transactional integrity, safeguarding data against loss or corruption. Furthermore, the user interface maintains responsiveness and visual consistency across multiple screen resolutions, ensuring a seamless experience for desktop. Table 4 presents a detailed summary of the module-level testing outcomes conducted on the StreamCats platform, providing a clear overview of system functionality and reliability under various test conditions.

Table 3 Summary Modules Final Test

Tested Module	Result	Status
WebSocket Connection	*Real-time* receive event	Passed
Event Handling	Event Handled correctly	Passed
Sound Alerts	Trigger play sound correctly	Passed
RCON Minecraft	Action correctly Executed	Passed
UI Responsiveness	Consistent with all resolution	Passed
Supabase Database	Data valid and safe	Passed

The validation on table 3 shows that all tested components passed, demonstrating the system's reliability and readiness for real-world deployment.

All modules passed functional validation, demonstrating the system's robustness and reliability for real-world deployment. In particular, the stability of Supabase aligns with the findings of Baccour et al. (2021), which emphasize that cloud-based databases can sustain scalable performance under distributed and concurrent loads (Septiadi & Rahmawati, 2023).

Building on this foundation, a usability evaluation was conducted with ten participants including students, content creators, and general users familiar with live streaming platforms covering tasks such as configuring events, triggering sound notifications, and monitoring real-time viewers that can be seen on Table 5.

Table 4 Participant Website Test

Subject	Interface Navigation	Clarity of Buttons & Icons	Time to Understand Each Feature	Task Completion
U1	Easy	Clear	4 min	Completed
U2	Easy	Clear	3 min	Completed
U3	Moderate	Clear	6 min	Completed
U4	Easy	Clear	4 min	Completed
U5	Easy	Clear	5 min	Completed
U6	Easy	Clear	5 min	Completed
U7	Moderate	Moderate	14 min	Not Completed
U8	Easy	Clear	4 min	Completed
U9	Easy	Clear	3 min	Completed
U10	Moderate	Clear	6 min	Completed

After analyzing participant responses at Table 5 interactions, observations indicate that 70% of participants found the interface easy to navigate, the majority considered buttons and icons sufficiently clear, task comprehension time averaged under five minutes, and 90% of users successfully completed all tasks. It can be seen on Table 6.

Table 6 Usability Test Summary

Evaluation Aspect	Evaluation Result
Interface Navigation	70% stated it is easy to navigate
Clarity of Buttons and Icons	Majority stated they are sufficiently informative
Time to Understand Each Feature	Average of less than 5 minutes
Task Completion Success	90% of respondents completed all tasks

Table 6 findings suggest that the user interface, designed with Tailwind CSS, is intuitive and consistent across devices, reducing the need for extensive user training (Wijaya & Hartono, 2020).

Despite these positive outcomes, certain limitations are noted. Testing did not cover poor network conditions, so system performance under high latency or packet loss remains uncertain. Furthermore, TikTok integration relies on reverse engineering rather than an official API, which may require ongoing maintenance to adapt to protocol changes. Nevertheless, the combined results of stress testing, module validation, and usability evaluation demonstrate that StreamCats is capable of supporting complex, multi-platform interactive streaming, offering a stable, efficient, and user-friendly experience. The system's architecture, leveraging modern web technologies and cloud services, ensures scalability, maintainability, and real-time responsiveness, making it competitive against existing platforms while providing a strong foundation for future enhancements such as AI-driven personalization and adaptive streaming features.

CONCLUSION

This study demonstrates that the StreamCats platform effectively addresses the challenges of latency, scalability, and interactivity in live streaming by integrating modular WebSocket communication, Supabase cloud storage, and AI-assisted features. Stress testing showed that latency remained well below the 5,000 ms threshold, averaging 2,180.3 ms even under 100 concurrent users, while CPU utilization stayed under 1% and throughput reached 89.8 requests per second, over 50% higher than Indofinity and more than four times Tikfinity. Usability testing further revealed that 90% of participants completed all tasks successfully, with an average comprehension time of under five minutes, confirming the platform's accessibility and ease of use.

Beyond these technical achievements, the findings highlight broader implications for both research and practice. Scientifically, the results confirm prior work suggesting that cloud-assisted, event-driven architectures can substantially reduce system load while improving responsiveness, extending this knowledge to third-party live streaming integration. Practically, StreamCats demonstrates that a lightweight, browser-based platform can empower streamers to deliver interactive experiences without requiring high-end devices or official APIs, thereby lowering entry barriers for content creators.

Future work may extend StreamCats with AI-driven personalization, adaptive streaming under unstable networks, and integration with additional platforms beyond TikTok. Overall, the system not only fulfills its stated objectives but also contributes to the ongoing development of scalable, cloud-enabled live streaming solutions that balance technical efficiency with user-centered design.

REFERENCES

Alharbi, A. A., Alqahtani, A. A., & Zainal, A. (2021). Enhancing maintainability in large-scale web applications using TypeScript. *IJACSA*, 12(9), 525–531. https://doi.org/10.14569/IJACSA.2021.0120964

Al-Msie'deen, R. (2018). Automatic labeling of the object-oriented source code: The Lotus approach. *Journal of Software Engineering and Applications, 18*(1), 1–15. https://doi.org/10.48550/arXiv.1803.00048

Baccour, E., Haouari, F., Erbad, A., Mohamed, A., Bilal, K., & Guizani, M. (2021). An intelligent resource reservation for crowdsourced live video streaming applications in geo-distributed cloud environment. *arXiv*. https://doi.org/10.48550/arXiv.2106.02420

Bokhari, H., & Herraiz, I. (2024). Security challenges and solutions in cloud models: IaaS, PaaS, SaaS. *Korea Science Journal of Information Security*.

Farahani, R., Timmerer, C., & Hellwagner, H. (2024). Towards low-latency and energy-efficient hybrid P2P-CDN live video streaming. *arXiv*. https://doi.org/10.48550/arXiv.2403.16985

Fedianto, M. H. S., Aditiawan, F. P., & Al Haromainy, M. M. (2023). Pengujian Sistem Jaringan Dokumentasi Dan Informasi Menggunakan Black Box Testing Dan White Box Testing. *Jurnal Publikasi Sistem Informasi Dan Manajemen Bisnis*, *3*(1), 213–221. https://doi.org/10.55606/jupsim.v3i1.2447

Hermawan, R., & Alim, S. (2021). Penerapan metode validasi fungsional pada pengembangan perangkat lunak web-based. *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK)*, 8(3), 411–418. https://doi.org/10.25126/jtiik.v8i3.6865

Huang, X., Li, M., Wu, W., Zhou, C., & Shen, X. S. (2022). Digital twin-assisted collaborative transcoding for better user satisfaction in live streaming. *arXiv*. https://doi.org/10.48550/arXiv.2211.06906

Husna, M., Maulana, R. M., & Sugiarti. (2024). Object-oriented analysis in software engineering: A systematic review of the literature. *Informatics and Software Engineering*, 2(2), 52–60.

Irawan, E., Rosjanuardi, R., & Prabawanto, S. (2024). Promoting computational thinking through programming: Trends, tools, and educational approaches—a systematic review. *Journal of Educational Technology & Society*, *27*(4), 45–59. 10.31764/jtam.v8i4.26407

Karunamoorthy, G., & Prakash, P. (2021). Unified process model for software development. *International Journal of Software Engineering and Applications (IJSEA)*, 12(2), 21–33. https://doi.org/10.5121/ijsea.2021.12202

Khder, A. (2021). Web scraping for data extraction: Challenges and solutions. *International Journal of Computer Science and Information Technology Research*.

Kumar, R. (2024). Cloud-based video streaming services: Trends, challenges, and opportunities. *CAAI Transactions on Intelligence Technology*. https://doi.org/10.1049/cit2.12299

Liu, X., & Zhang, L. (2024). Impacts of different interactive elements on consumers' purchase intention in live streaming e-commerce. *PLOS ONE*, *19*(12), e0315731. https://doi.org/10.1371/journal.pone.0315731

Mardiah, A., Evanita, S., & Syawalki, L. (2023). Interactive live streaming: Analysis of online marketing communication in online shopping. *ProBisnis: Jurnal Manajemen*, 14(4), 104–109. https://doi.org/10.62398/probis.v14i4.246

Maulidina, N., & Nugraha, R. (2023). Evaluasi performa WebSocket dalam komunikasi real-time pada aplikasi web. *Jurnal Teknologi Informasi dan Komputer*, 9(2), 145–152.

Mei, L., Tang, N., Zeng, Z., & Shi, W. (2025). Artificial intelligence technology in live streaming e-commerce: Analysis of driving factors of consumer purchase decisions. *International Journal of Computers Communications & Control*, 20(1), 6871. https://doi.org/10.15837/ijccc.2025.1.6871

Mishra, A., & Mishra, A. (2024). The research on trust in leadership: The need for context. *Journal of Organizational Behavior*, *45*(2), 234–250. https://doi.org/10.1080/21515581.2013.771507

Prihartono, W., & Utami, S. F. (2023). Rational unified process as a software development method in decision support systems. *Jurnal Sisfotek Global*, *13*(2), 123–135. http://dx.doi.org/10.38101/sisfotek.v13i2.9678

Septiadi, B. P., & Rahmawati, Y. (2023). Analisis validitas data pada cloud database menggunakan Supabase dan PostgreSQL. *Jurnal Sistem dan Teknologi Informasi*, 11(2), 78–85.

Siala, H. A., Lano, K., & Alfraihi, H. A. (2024). Model-driven approaches for reverse engineering: A systematic literature review. *IEEE Access*, 12(99), 62558–62580. https://doi.org/10.1109/ACCESS.2024.3394732

Wang, L., He, Y., Ding, J., Huang, N., Hong, Y., Guo, X., Liu, D., & Chen, G. (2021). Effectiveness of AI assistance in live streaming: A randomized field experiment. *Proceedings of the 42nd International Conference on Information Systems (ICIS 2021)*. https://aisel.aisnet.org/icis2021/hci_robot/hci_robot/3/

Wang, L., Huang, N., Hong, Y., Liu, L., Guo, X., & Chen, G. (2023). Voice-based AI in call center customer service: A natural field experiment. *Production and Operations Management*, *32*(4), 1002–1018. http://dx.doi.org/10.1111/poms.13953

Wijaya, H., & Hartono, E. (2020). Analisis kinerja dan kualitas sistem interaktif berbasis streaming menggunakan metode empiris. *Jurnal Sistem Informasi*, 16(1), 42–51. https://doi.org/10.21609/jsi.v16i1.882

Wijaya, Y. D. (2021). Pengujian Blackbox Sistem Informasi Penilaian Kinerja Karyawan PT Inka (Persero) Berbasis Equivalence Partitions. *Jurnal Digital Teknologi Informasi*, *4*(1), 22–26. https://doi.org/10.32502/digital.v4i1.3163

Xie, Y., & Desouza, K. C. (2025). Feeling grounded when watching live streaming shows of highly anthropomorphic interactive virtual influencers: An exploratory study on customer opinions. https://doi.org/10.1016/j.jbusres.2025.115507

Yu, S.-Y., Achamyeleh, Y. G., Wang, C., Kocheturov, A., Eisen, P., & Al Faruque, M. A. (2023). Cfg2vec: Hierarchical graph neural network for cross-architectural software reverse engineering. In *Proceedings of the 45th International Conference on Software Engineering (ICSE 2023)* (pp. 281–291). IEEE. https://doi.org/10.1109/ICSE-SEIP58684.2023.00031