Quiz Quest: An Open-Source Ephemeral Quiz Application For Efficient and Low Cost Interactive Education

Yustus Eko Oktian[1*], Evelin Alim Natadjaja[2], Evelyn Komalasari Hartono[3]

[1,2,3]Department of Information Systems, School of Information Technology, Universitas Ciputra, Surabaya, East Java, Indonesia
E-mail: [1*]yustus.oktian@ciputra.ac.id, [2]enatadjaja@student.ciputra.ac.id, [3]ekomalasari@student.ciputra.ac.id

(Received: 18 Sep 2025, revised: 16 Oct 2025, accepted: 17 Oct 2025)

Abstract

The increasing integration of technology in educational settings has highlighted the efficacy of interactive tools, such as online quizzes, in fostering student motivation and engagement. However, the widespread adoption of commercial platforms is often impeded by substantial financial and logistical challenges, including high costs and complex manual setup procedures. To address these limitations, this paper presents Quiz Quest, a novel open-source, ephemeral quiz application. Quiz Quest is engineered using contemporary web service development technologies and features a streamlined, JSON-based question import mechanism, enabling educators to host interactive quizzes on their personal devices. Preliminary assessments indicate that Quiz Quest offers both efficiency and competitive performance. Specifically, compared to established alternatives like Kahoot and Google Forms, Quiz Quest demonstrates a reduction in question setup time of 67-74% during initial use and 71-83% in subsequent uses. Furthermore, it exhibits minimal system resource consumption in low-density classroom environments, with CPU utilization ranging from 0.05% to 0.1% and memory usage stabilizing around 90-95 MB, thereby ensuring compatibility across a variety of hardware configurations. The application also achieved a favorable System Usability Scale score of approximately 75, performing comparably to its counterparts. Consequently, Quiz Quest is posited as an accessible, efficient, and engaging solution for interactive education, offering significant benefits to educators seeking to enhance learning experiences without incurring considerable resource expenditure.

Keywords: Interactive Education, Online Quizzes, Educational Technology, Software Engineering

I. INTRODUCTION

Research indicates that integrating technology into educational settings can enhance student motivation and engagement through interactive tools like online quizzes [1] [2]. Gamified platforms such as Kahoot and Quizizz have been shown to significantly increase student motivation and engagement [2] [3], and improve learning outcomes, as evidenced by higher post-test scores and better conceptual retention [4] [2]. Consequently, the adoption of these cost-effective gamified tools has led to measurable academic achievement gains in various university courses [5] [2].

Despite the motivational benefits of online quizzes, modern applications can be costly. For instance, Kahoot [2] increasingly directs users toward paid subscriptions, posing a significant barrier for budget-constrained educational institutions. Currently, its free tier is limited to 15 participants, a reduction from previous years. Furthermore, setting up new quizzes on these platforms can be time-consuming, often requiring manual entry of questions and options. While Kahoot offers AI features, full utilization is restricted to paid versions.

In response to these challenges, we introduce Quiz Quest, an open-source, ephemeral quiz application designed for efficient and low-cost interactive education, addressing the limitations of current commercial offerings. Our application leverages modern web service development tools, allowing instructors to host quizzes on their personal devices, with students joining in real-time. A key feature is the ability to import questions via a JSON-based text file, facilitating rapid deployment without manual input.

Our evaluations demonstrate that Quiz Quest significantly reduces preparation time for educators compared to alternatives like Kahoot [2] and Google Forms [6]. The application also achieves a competitive System Usability Scale score relative to these platforms. Combined with its ephemeral hosting and open-source nature, Quiz Quest offers an accessible and engaging solution for interactive education.

The remainder of this paper is structured as follows: Section 2 reviews existing online quiz platforms. Section 3

DOI: 10.34148/teknika.v14i3.1383

CC BY SA

details the system model and internal workings of our application. Section 4 presents the evaluation of experimental and usability results. Section 5 discusses limitations and potential future improvements. Finally, Section 6 concludes the paper.

II. RELATED WORK

The field of educational technology includes many efforts to create interactive quiz applications and platforms. Popular commercial solutions like Kahoot and Quizizz are known for their engaging, gamified quizzes that allow real-time interaction and access across devices [2]. Research on the "Development of the Quizizz Platform" [4] highlights how interactive quizzes can motivate learners with attractive designs and help assess understanding. Similarly, a study on "Developing gamification e-quizzes based on an Android app" [7] focuses on a mobile-first approach for self-paced learning and evaluating students using game-like features. Another relevant study, "The Development of Evaluation Instruments in Online Learning using the Quizizz Application" [8], discusses creating and validating assessment tools for online learning, showing how existing platforms can be adjusted for specific teaching needs.

However, there is a lack of truly open-source, affordable, and resource-friendly interactive quiz applications that can be used in various educational settings without causing financial or technical difficulties. While the Quizizz platform [4] is accessible and motivating, it is a commercial product that may have subscription fees and limitations. The Android gamified e-quizzes [7], although accessible on mobile and good for self-paced learning, are specific to that platform and might not offer the same flexibility for live hosting by instructors using their own devices without complicated server setups. Furthermore, efforts to create assessment tools with Quizizz [8] depend on the commercial platform, sharing its costs and operational requirements. Many current projects focus on commercial growth, specific device types, or integration with complex databases, which can increase costs, technical needs, or limit adaptability for quick classroom use. Thus, there is a clear need for a solution that is open source, can be used temporarily, uses few resources, and simplifies question setup for teachers.

Table 1. Feature Comparison Between Our Proposed Application (Quiz Quest) and the Alternatives

Features	Quiz Quest	Kahoot	Google Forms
Open-Source	Yes	No	No
Cost-Effective	High	Low	High
Question Import (JSON/CSV)	Yes	Yes	No
Real-Time Interaction	Yes	Yes	No
Gamification	Yes	Yes	No
BYOD Hosting	Yes	No	No
Persistent Data Storage	No	Yes	Yes

Unlike previous efforts and commercial options, Quiz Quest is an innovative open-source quiz application that can be used ephemerally. It stands out with a simple JSON-based feature for importing questions, greatly reducing the time teachers spend preparing—a feature often only available in paid versions or missing entirely in other apps. Moreover, Quiz Quest is designed to use very few resources, allowing teachers to host quizzes on their own devices without needing costly equipment or regular payments. This approach directly tackles the cost barriers, setup difficulties, and resource demands of many current interactive educational tools, offering an accessible, efficient, and engaging solution for dynamic learning. Feature comparisons between our proposed application and the alternatives can be seen in Table 1.

III. SYSTEM MODEL

This section elaborates the internal mechanisms of our application, detailing its feature composition, software architecture, workflow, and data storage.

A. Feature Composition

We first outline several design decisions implemented to address the shortcomings of existing quiz platforms and ensure the usability and efficiency of our application.

JSON Text Import: While many real-time quiz applications support question importing, often through CSV files, for rapid quiz setup, this feature is frequently placed behind a paywall. Consequently, an import functionality is deemed essential for our proposed application's competitiveness. Our design facilitates JSON text import, offering greater data expression flexibility than CSV. To mitigate potential difficulties for non-developer users with JSON format, we integrate an out-of-band AI service for assistance in generating compatible JSON text.

Bring Your Own Device: In alignment with our open-source objective, we enable instructors to utilize their own hardware for hosting quizzes. To circumvent network accessibility issues arising from typical institutional NAT configurations, which often impede external access, we employ public port forwarders like Serveo. This bypasses the need for public IP addresses, thereby avoiding additional financial investment. Furthermore, we provide a Docker image on Docker Hub, simplifying application deployment through a single Docker command.

Interactive and Fun Experience: To enhance user engagement, we incorporate gamification elements similar to other quiz platforms. These include scoring based on response speed, correct answer streaks, leaderboards, quiz statistics reports, and customizable scoring and time limits per question, allowing for more expressive and dynamic question scoring.

Web socket Connection: To ensure interactivity, rapid feedback from the server is crucial. This is achieved through bidirectional communication via web sockets. Our aim is to establish robust web socket connections between clients and instructors, addressing potential failure scenarios such as

connection loss, disruptive student behavior, or network interruptions.

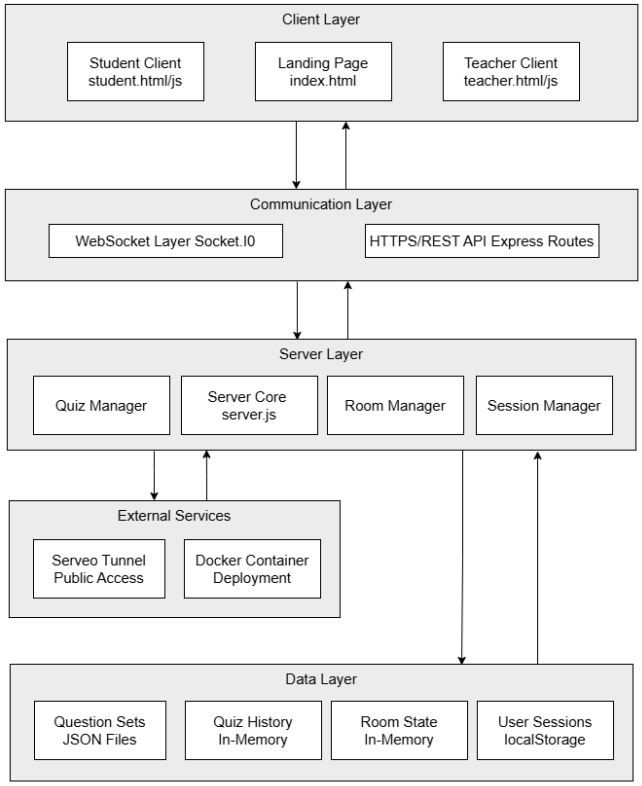


Figure 1. Architecture Diagram of Our Proposed System Showing Five Major Components: Client Layer, Communication Layer, Server Layer, Data Layer, and External Services.

B. System Architecture

Based on an analysis of required features, the application is designed around a client-server model, comprising five distinct components as illustrated in Figure 1.

- **Client Layer:** This layer encompasses three primary components: the landing page, the client view, and the teacher view.
- **Server Layer:** Functioning as the central operational hub, this layer manages core logic, data processing, inter-client communication, and interactions with external services.
- **Communication Layer:** This layer facilitates real-time, bidirectional data exchange between clients and the server using web sockets. It also exposes several endpoints via REST API for clients requiring updated server data.
- **Data Layer:** Responsible for the persistence and retrieval of quiz-related data, including questions, answers, participant scores, and session states, this layer utilizes a combination of in-memory databases and the browser's local storage.
- **External Services:** This layer manages dependencies on external services, such as public port forwarding for internet accessibility and Docker services for easy access to the latest application images from Docker Hub.

C. Process Workflow

The following outlines the general workflow of our application, from teacher login to viewing quiz session history.

Teacher Authentication: The application features distinct interfaces for teacher and student roles. Students can join and participate in quizzes without authentication, whereas teachers must log in to their interface using a password. This measure is in place to prevent unauthorized access to the admin console and potential disruption of the quiz flow by malicious students.

Teacher Creates Questions: Teachers can utilize an out-of-band AI system to generate quiz content for the application. The teacher dashboard provides a prompt that, when used with the AI system, generates JSON text compatible with our app. This AI-assisted quiz creation capability significantly simplifies and expedites the quiz setup process.

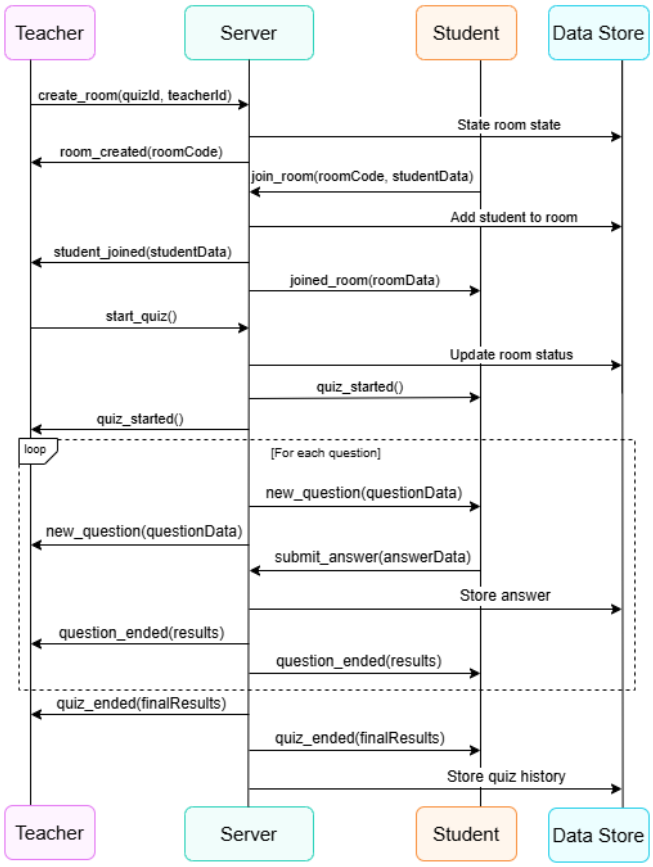


Figure 2. Sequence Diagram of Quiz Room Life Cycle Starting from Creating Room, Starting Quiz, and Saving Quiz History

Teacher Runs the Quiz: Once questions are imported, they appear on the teacher dashboard. Teachers can then create a quiz room (one question set at a time) and generate a random room ID that students can use to join via the student view's quiz join room interface. The room ID is presented in three formats: a numerical code for manual input, a URL for easy sharing, or a QR code for quick scanning.

DOI: 10.34148/teknika.v14i3.1383
TEKNIKA, Volume 14(3), November 2025, pp. 509-517
ISSN 2549-8037, EISSN 2549-8045

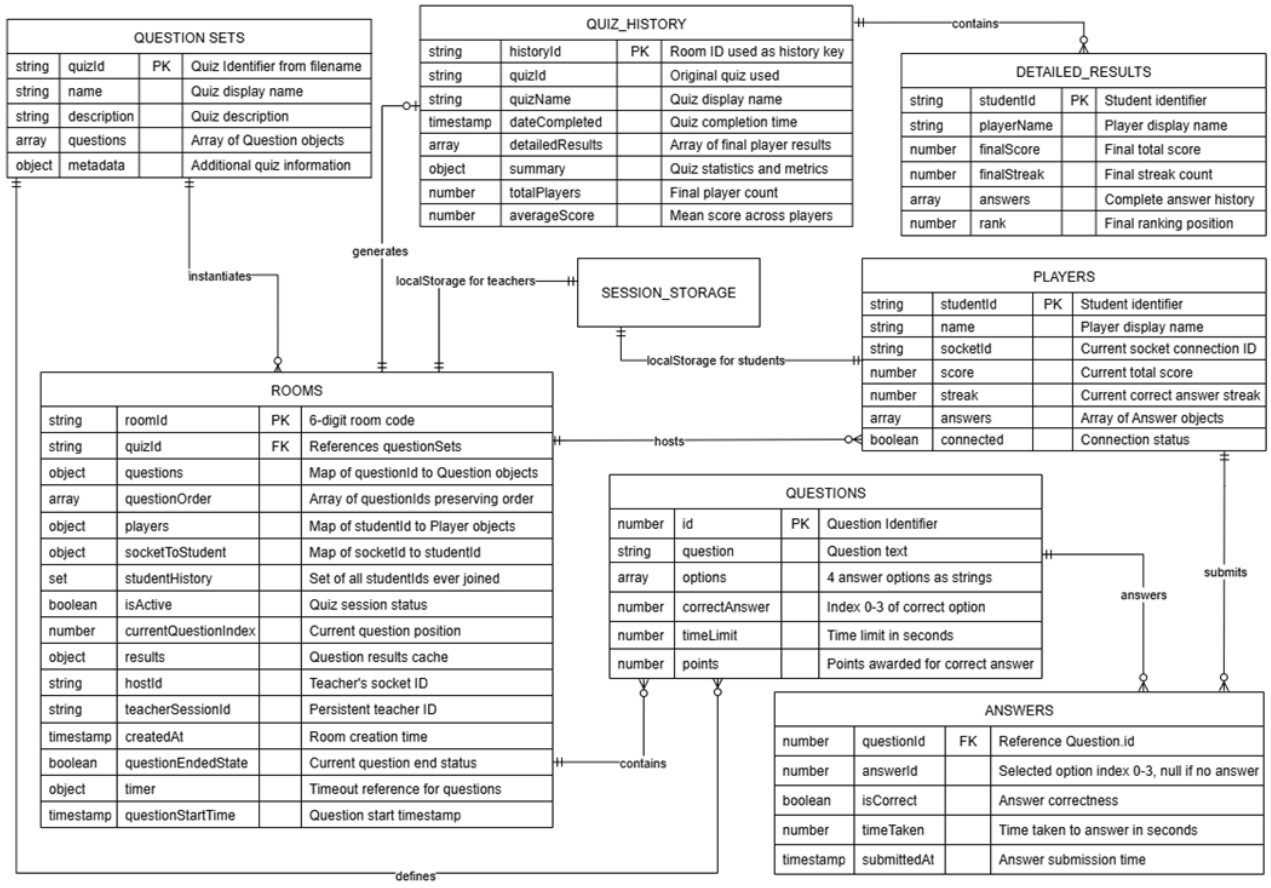


Figure 3. The Data Store Schema for Our Proposed System.

In the waiting room, both teachers and students can monitor the number of participants. When the teacher deems all students have joined, the quiz can commence. The server then displays the questions to both teacher and student interfaces. Students must answer within the time limit, after which the correct answer is revealed, and scores are updated. Upon quiz completion, the final student rankings are presented. Teachers have the option to download these rankings in CSV format to their local storage. This entire quiz operation is summarized in Figure 2.

As a supplementary measure, during the quiz, the browser's local storage is employed to preserve quiz states, ensuring data persistence across sessions and disconnections for both teachers and students.

Teacher Trace Back the Quiz Session: For instances where a teacher might forget to save or download previous quiz scores, the History menu provides access to completed room sessions. From the detailed history list, teachers can review the number of participants and the final rankings, and again download the final rankings in CSV format. The downloaded CSV file includes not only final scores but also logs of individual student answers for each question, proving useful for analytics, such as identifying questions that posed difficulties for students. It is important to note that as this data is not stored in a persistent database, it is lost upon server shutdown.

D. Data Storage

The data storage is architected to comprise seven objects: question sets, rooms, quiz history, players, answers, detailed results, and questions. The detailed relationships between these objects are visually represented in Figure 3. Notably, the datastore is distributed across three mediums: in-memory datastore, browser's local storage, and the teacher's local storage.

IV. EXPERIMENTAL RESULTS

This section presents a discussion of several experimental results aimed at analyzing the proposed system. Initially, the implementation of the system is showcased. Subsequently, a performance overhead analysis is conducted, focusing on CPU and memory utilization. Following this, a comparative analysis of setup time requirements is elaborated, contrasting our application with Kahoot and Google Forms. Kahoot was selected as a representative of modern, interactive quiz applications, while Google Forms was chosen to represent traditional applications lacking interactivity. Throughout this experiment, we utilize the free tier version of Kahoot and Google Forms. Finally, the usability of the proposed application is evaluated.

CC BY SA

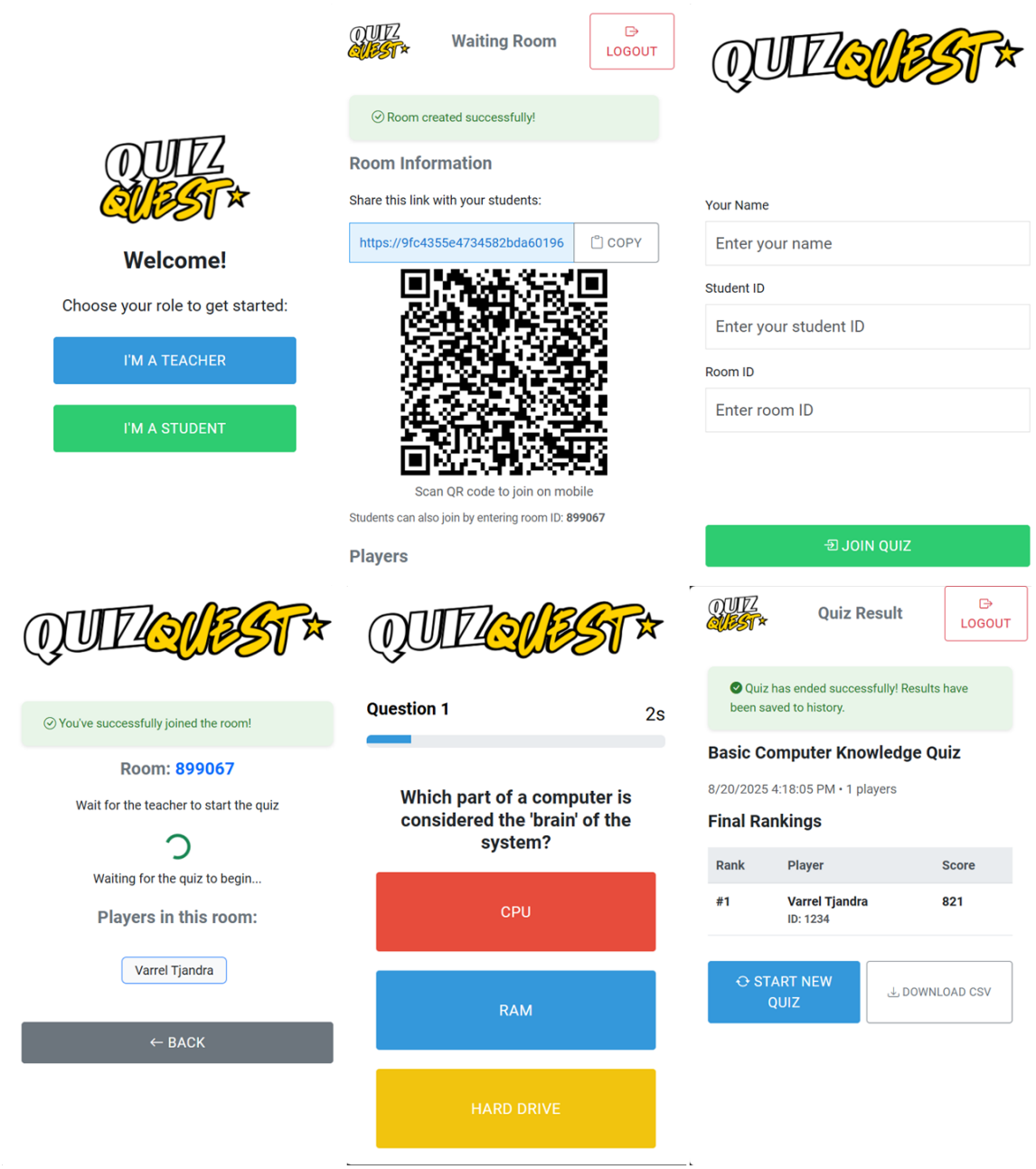


Figure 4. User Interface for Our Proposed System

A. Implementation

Our proposed system is primarily implemented using Node.js [9] on the server-side, leveraging Express for REST-API endpoints and Socket.io [10] for real-time web sockets connections that enable student interaction during quizzes. For the client side, we utilize basic HTML, CSS, and JavaScript for both the teacher and student interfaces. Figure 4 displays the user interface of our application, showing from top left to bottom right: the landing page, the teacher's quiz creation and waiting room, the student's join form, the student's waiting room, the question display, and the quiz results. Throughout our evaluation, the experiment was conducted on a laptop with specifications including an Intel Core i5-8250U CPU @1.60GHz and 16GB of RAM.

B. System Overhead

We first analyze the resource requirements of the application when running in a class setup environment with one teacher and three students. For this setup, we ran our application on a Docker environment [11]. First, we launched the application and then measured the CPU and memory usage of the container using Cadvisor, Prometheus, and Grafana stack [12]. The results are summarized in Figure 5.

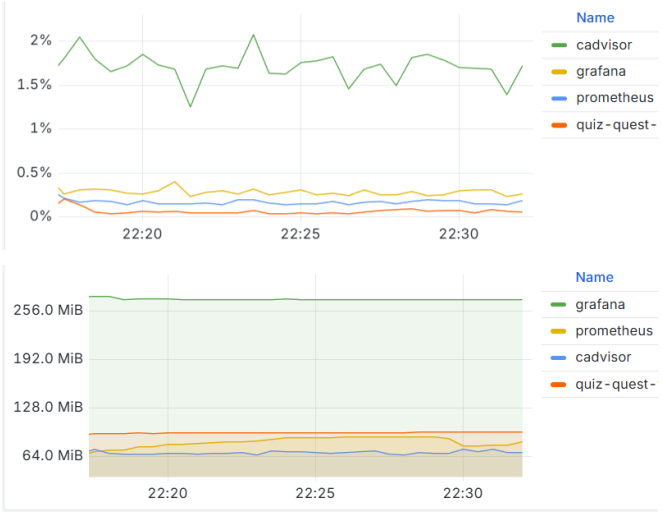


Figure 5. The Recorded CPU (Up) and Memory Usage (Bottom) for Our Proposed System (Quiz Quest).

The data presented in Figure 5 indicates that CPU utilization ranges from 0.05% to 0.1% over a 15-minute operational period, with memory consumption stabilizing around 90-95 MB. These findings confirm that the resource demands of Quiz Quest are notably modest, thus supporting its suitability for a diverse array of devices with minimal impact on performance.

C. Setup Time Measurement

We ask 10 respondents from 5 different institutions ranging from K-12 to higher education to test our proposed application. First, they complete a pre-test quiz to gather background information, including teaching experience, familiarity with online quiz applications, frequency of use, and their working environment (laptop hardware, internet reliability). Detailed information about respondent demography can be seen in Appendix.

Second, we visit them onsite and instruct them to perform two tasks using Quiz Quest, Kahoot, and Google Forms. Their completion times are meticulously recorded to quantify efficiency gains. The tasks are conducted multiple times: an initial run without guidance, followed by a 10-minute exploration and study period, and then a second run of the same tasks. To prevent bias, the order of application execution (first, second, third) is randomized across participants.

Task 1: Teachers are provided with a PDF file containing multiple-choice questions to input into each platform. The elapsed time is measured from the start of the instruction until the room code appears on the screen. For Google Forms, we measure the time up to the form is ready to publish with the link.

Task 2: Teachers are also asked to set up or bootstrap the application. For our application, this involves opening the terminal, running the Docker command, and opening a web browser to display the dashboard. The elapsed time is measured from the instruction to the landing page appearing in the browser. For Kahoot and Google Forms, participants open the respective landing pages, and the elapsed time is measured.

The result of this experiment from Task 1 and 2 for all runs are presented in Figure 6.

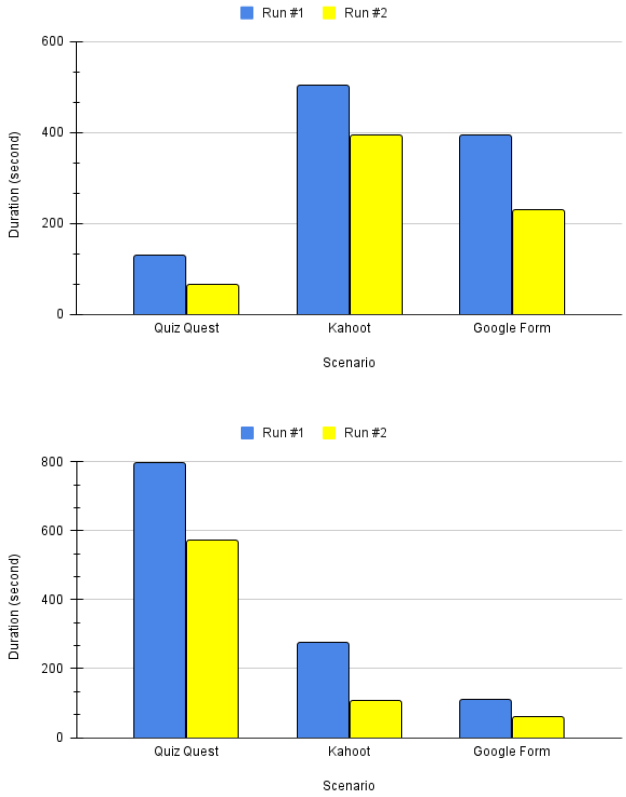


Figure 6. The Average Setup Time Required Per Question for Task 1 (Up) and per Application for Task 2 (Bottom) for All Runs Between Our Proposed System (Quiz Quest) and Alternatives (Kahoot and Google Forms).

Analysis of Figure 6 indicates a trend where participants experienced initial difficulties performing the tasks without prior guidance, as evidenced by the reduction in completion time during their second attempt across all application scenarios. This suggests an evident learning curve associated with each platform, underscoring the influence of user interface intuitiveness and prior experience on initial setup complexities in educational technology.

Our proposed system demonstrates significant efficiency gains, reducing setup time by 67-74% in the first run and 71-83% in the second run compared to alternative platforms. This improvement is attributed to its capability to import JSON text and its integration with an out-of-band AI service prompt for JSON generation, thereby streamlining the process.

Conversely, Task 2 revealed challenges, with our system requiring 2.9 to 7.2 times longer in the first run and 5.3 to 9.7 times longer in the second run for users to manually execute the application via the Command Line Interface (CLI). These findings highlight a potential unfamiliarity with CLI execution among respondents and underscore a critical area for future development.

D. Usability Metric

After the completion of Tasks 1 and 2, participants were administered a ten-item System Usability Scale (SUS) questionnaire [13], [14], with further details provided in the Appendix. Respondents completed the SUS questions on two occasions: once following Run 1 and again after Run 2.

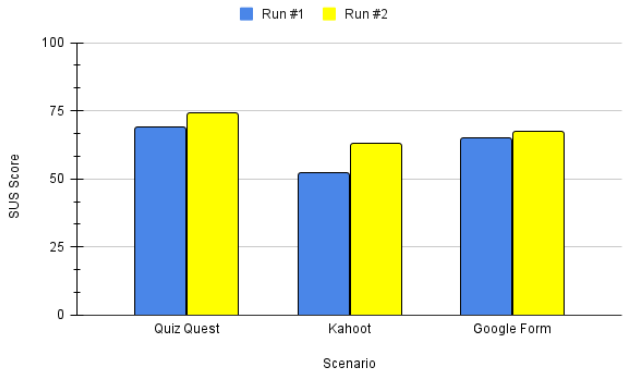


Figure 7. Comparison of SUS Score Between Our Proposed System (Quiz Quest) and the Alternatives (Kahoot and Google Forms).

The results of the System Usability Scale scoring are depicted in Figure 7. Our proposed system achieved an average SUS score of approximately 75, signifying a favorable level of usability that aligns with recognized benchmarks for acceptable system performance. In comparison, Kahoot and Google Forms scored below this benchmark, suggesting potential areas for improvement in their user interface or functionality to enhance user-friendliness. Throughout the experiment, a clear trend indicated that as participants became more familiar with the application, their satisfaction and perceived ease of use improved, suggesting that prolonged usage could further elevate the SUS scores.

V. DISCUSSION

This section elaborates on the weakness of our current applications and derives possible improvements for the future work.

First, our application currently runs on a Docker environment. This necessitates the host machine to have Docker installed and configured, which may present a technical barrier for less experienced users who are not familiar with operating a CLI terminal. This highlights a trade-off between cost and usability. Deploying this application in a cloud environment would undoubtedly enhance usability but also incur higher costs. Alternatively, we could explore an on-premises approach where an institution hosts the application on its own server, though this also entails additional capital or operational expenditure. For future work, we can develop this app with an executable binary so that it can be easily installed and run by average users.

Second, in the current iteration of application development, we use vanilla HTML, CSS, and Javascript code to build this application. Switching to a modern framework such as React or Svelte would significantly improve the maintainability, scalability, and developer experience of the application [15]. Furthermore, adopting these frameworks would allow for more sophisticated user interface components and a more responsive, dynamic user experience, aligning with contemporary web application standards.

Third, the design philosophy of our application is to create ephemeral quiz sessions. As a result, we do not utilize any database in the current iteration of the application. If instructors forget to download the quiz results, there is no way for them to obtain the previous quiz session data. Further improvements can be made by incorporating a local database, such as using MySQLi or a MySQL database [16].

Fourth, the application lacks audit and security features. Given its open-source nature, this creates vulnerabilities for malicious actors to develop cheat systems. Additionally, the absence of robust user authentication and authorization mechanisms permits fake participants, potentially allowing external individuals to participate in quizzes on behalf of others. These vulnerabilities highlight areas for future work focused on enhancing the application's security [17].

Finally, this study was conducted as a small-scale exploratory evaluation involving ten educators from various educational levels. While this design provides useful insights into the system's feasibility and usability, the limited sample size and convenience-based selection restrict the generalizability of the results. Participants' prior familiarity with Kahoot and Google Forms may have given them an advantage when performing tasks on those platforms compared to our newly introduced system. Nevertheless, despite this familiarity bias, the proposed system achieved competitive setup times and usability scores, indicating that its interface and workflow were intuitive even for first-time users. Environmental factors such as hardware specifications, network stability, and novelty effects could also have influenced timing and satisfaction outcomes. These limitations are acknowledged to contextualize the findings as preliminary yet informative for future large-scale evaluations.

VI. CONCLUSION

This paper introduced Quiz Quest, an innovative open-source and ephemeral quiz application developed to overcome the significant financial and logistical hurdles associated with commercial interactive educational platforms. By leveraging modern web service technologies and a streamlined JSON-based question import feature, Quiz Quest demonstrably reduces educators' preparation time and achieves competitive usability metrics, as indicated by its favorable System Usability Scale scores. Our evaluations confirmed its remarkably low resource consumption, ensuring its accessibility and viability across diverse devices. Quiz Quest thus provides an accessible, efficient, engaging, and cost-effective solution for dynamic, real-time learning experiences. For future enhancements, we aim to bolster security features

DOI: 10.34148/teknika.v14i3.1383
TEKNIKA, Volume 14(3), November 2025, pp. 509-517
ISSN 2549-8037, EISSN 2549-8045

and integrate persistent data management capabilities to our prototype.

REFERENCES

[1] J. Guaña–Moya, Y. Arteaga-Alcívar, S. Criollo-C, and D. I. C. Carrazco, "Use of Interactive Technologies to Increase Motivation in University Online Courses," *Education Sciences* , vol. 14, no. 12, p. 1406, Dec. 2024, doi: 10.3390/educsci14121406.

[2] B. Maraza-Quispe, "Impact of the Use of Gamified Online Tools: A Study with Kahoot and Quizizz in the Educational Context," *International Journal of Information and Education Technology* , vol. 14, no. 1, p. 132, Jan. 2024, doi: 10.18178/ijiet.2024.14.1.2033.

[3] D. Dolezal, A. Posekany, R. Motschnig, and R. Pucher, "Effects of Introducing a Game-Based Student Response System into a Flipped, Person-Centered Classroom on Object-Oriented Design," in *Lecture notes in computer science* , Springer Science+Business Media, 2018, p. 132. doi: 10.1007/978-3-319-96565-9_13.

[4] A. Hanif, Herman, A. Mudinillah, and P. W. L. Rahmi, "Development of the Quizizz Platform as an Interactive Quiz-Based Learning Media for Arabic Language Lessons at Madrasah Ibtidaiyah Akhyar Hanif1*, Herman2, Adam Mudinillah3, Putri Windi Lailatur Rahmi4." 2023.

[5] K. Altawalbeh, "Game-Based Learning: The Impact of Kahoot on a Higher Education Online Classroom," *Journal of Educational Technology and Instruction.* , vol. 2, no. 1, p. 30, Mar. 2023, doi: 10.70290/jeti.v2i1.13.

[6] R. Saskia, M. Okuda, and B. Cooney, "Utilization of Google From as a Quiz for Learning Fiqh," *Scientechno Journal of Science and Technology* , vol. 2, no. 1, p. 49, Apr. 2023, doi: 10.55849/scientechno.v2i1.45.

[7] M. F. Areed, M. A. Amasha, R. A. Abougalala, S. Alkhalaf, and D. Khairy, "Developing gamification e-quizzes based on an android app: the impact of asynchronous form," *Education and Information Technologies* , vol. 26, no. 4, p. 4857, Mar. 2021, doi: 10.1007/s10639-021-10469-4.

[8] S. Haryati, S. W. Albeta, D. Futra, and A. D. Siregar, "The Development of Evaluation Instruments in Online Learning using the Quizizz Application: During Covid-19 Pandemic," vol. 13, no. 1, p. 364, May 2021, doi: 10.35445/alishlah.v13i1.383.

[9] M. Y. Ahyadi, "Development and Deployment of Crowd Monitoring System Using Nodejs and Redis on Infrastructure as a Service Model," p. 1, Jul. 2023, doi: 10.1109/icwt58823.2023.10335396.

[10] A. K. Das and S. Ghosh, "Real-Time Virtual Classroom Platform: Integrating React, Firebase, WebRTC, and Socket.io." 2025.

[11] B. Antony, "Containerization: Practical infrastructure and accessibility efficiency for the Virtual Learning Environment," *Pacific Journal of Technology Enhanced Learning* , vol. 2, no. 1, p. 41, Jan. 2020, doi: 10.24135/pjtel.v2i1.48.

[12] M. Dinga, I. Malavolta, L. Giamattei, A. Guerriero, and R. Pietrantuono, "An Empirical Evaluation of the Energy and Performance Overhead of Monitoring Tools on Docker-Based Systems," in *Lecture notes in computer science* , Springer Science+Business Media, 2023, p. 181. doi: 10.1007/978-3-031-48421-6_13.

[13] P. Vlachogianni and N. Τσέλιος, "Perceived usability evaluation of educational technology using the System Usability Scale (SUS): A systematic review," *Journal of Research on Technology in Education* , vol. 54, no. 3. Taylor & Francis, p. 392, Feb. 23, 2021. doi: 10.1080/15391523.2020.1867938.

[14] P. Y. Pratiwi, I. M. A. Pradnyana, and N. K. W. Damayanti, "Usability Analysis on Digital Library Information System using System Usability Scale (SUS)," p. 293, Sep. 2023, doi: 10.1109/ieit59852.2023.10335582.

[15] J. Tong, R. R. Jikson, and A. A. S. Gunawan, "Comparative Performance Analysis of Javascript Frontend Web Frameworks," Aug. 2023, doi: 10.1109/ice3is59323.2023.10335250.

[16] T. Capris, P. Melo, N. M. García, I. M. Pires, and E. Zdravevski, "Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation," in *2021 International Conference on Data Analytics for Business and Industry (ICDABI)* , Oct. 2022, p. 214. doi: 10.1109/icdabi56818.2022.10041513.

[17] D. V. Sheela and R. Chitra, "Securing Online Quiz Management through Blockchain Technology and Three-Tier Cryptography with Hash-based Authentication," p. 135, Oct. 2023, doi: 10.1109/i-smac58438.2023.10290272.

APPENDIX

This section comprises supplementary materials that offer further insights into the evaluation methodology, complementary of the primary evaluation section.

Respondent Demographics: A cohort of ten respondents, comprising educators from K-12 to university levels, participated in the application testing. This group included teachers from elementary, junior high, and senior high schools, as well as university lecturers, ensuring a diverse range of perspectives on quiz deployment across various educational settings. Detailed demographic distributions are presented in Table 2 and Figure 8.

Table 2. Respondent Demographics and Teaching Institutions

ID	Level	Institution Name
1	University	Universitas Ciputra
2	University	Universitas Ciputra
3	Senior High School	SMA Sekolah Ciputra Kasih
4	Senior High School	SMA Sekolah Ciputra Kasih
5	Junior High School	SMP Sekolah Citra Berkat
6	Junior High School	SMP Sekolah Citra Berkat
7	Junior High School	SMPK Untung Suropati
8	Junior High School	SMPK Untung Suropati
9	Junior High School	SMPK Untung Suropati
10	Elementary School	SDK Untung Suropati II

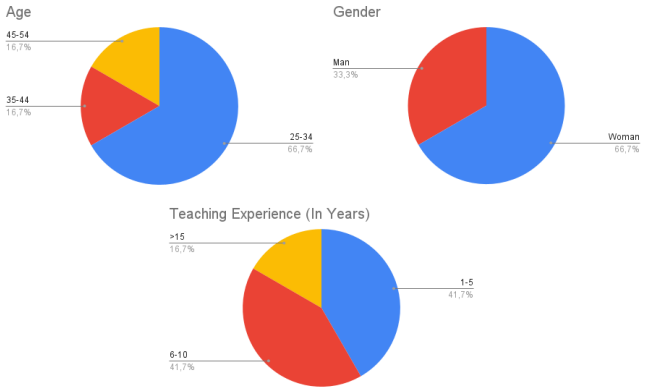


Figure 8. The Distribution of Respondents Based on Age, Gender, and Teaching Experience.

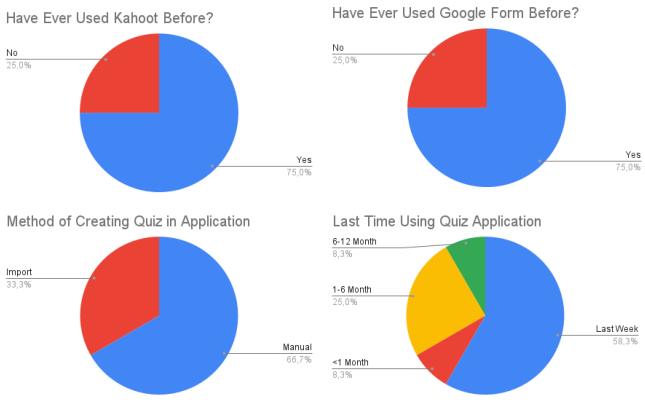


Figure 9. The Participants' Familiarity With Online Interactive Quiz Applications.

Familiarity with Online Quizzes: A survey question assessed respondents' prior experience with alternative quiz applications and their methods for quiz creation. As indicated in Figure 9, a significant majority (75%) of participants had previously used similar applications. Among these, 66.7% reported using a manual mode, which involves direct input via forms for questions and answers, while only one-third had experience with an import feature. Furthermore, 58.3% of respondents indicated last using such an application in the prior week, suggesting a general familiarity with online quiz tools. This existing familiarity is expected to minimize potential biases and reduce the learning curve associated with our application.

System Usability Scale (SUS) Questions: The SUS data was collected through a questionnaire featuring ten items (c.f., Table 3). Each item was assessed on a 5-point Likert scale, ranging from 'Strongly Disagree' to 'Strongly Agree.' The survey design intentionally included both positively and negatively worded statements to ensure robust data collection. Participants completed the questionnaire twice, once after completing Run 1 and again after Run 2.

Table 3. The Questionnaire Employed For The System Usability Scale Assessment.

ID	Questions
Q1	I think I will often use this application.
Q2	I find this application too complicated.
Q3	I find this application easy to use.
Q4	I think I need technical support to be able to use this application.
Q5	I feel that the various functions in this application are well integrated.
Q6	I find there is too much inconsistency in this application.
Q7	I imagine most people will quickly learn to use this application.
Q8	I find this application very cumbersome to use.
Q9	I am very confident that I can use this application smoothly.
Q10	I feel I need to learn many things before I can use this application.

Source Code: Our quiz application are developed using AI assistant, monitored, and controlled by the authors. The full source code is available at the following link: https://github.com/mrkazawa/quiz-quest-vibe

DOI: 10.34148/teknika.v14i3.1383
TEKNIKA, Volume 14(3), November 2025, pp. 509-517
ISSN 2549-8037, EISSN 2549-8045