Jurnal Ilmiah Edutic : Pendidikan dan Informatika /Vol.12, No.2 : 2025
ISSN 2407-4489
pp. 113-121

Development of a Web-Based Online Judge for Java Programming Examinations to Enhance Interactive Learning Environments

Muhammad Trio Maulana Putra[a], Nurul Laili Sa'adah[b], Nicky Dwi Kurnia[c], Febi Warta Nur Ani[d]

[a,b,c,d] *Information Technology Education, Institut Teknologi Mojosari, Nganjuk, Indonesia*
[b] *Informatics Engineering Education, Universitas Negeri Malang, Malang, Indonesia*
email:[a] triomaulana@itmnganjuk.ac.id,[b] nurullaili@itmnganjuk.ac.id nurul.laili.ft@um.ac.id,
[c]nickydwikurnia1994@gmail.com, [d]febiwarta.nurani@gmail.com

Abstract

This paper addresses the challenges in traditional programming assessment, such as scalability and delayed feedback. We present the development of a web-based examination platform for the Java programming language. The system is designed with a decoupled architecture, integrating the CodeMirror editor to create an interactive learning environment and leveraging the Glot.io API for secure, sandboxed code execution. This approach mitigates security risks associated with evaluating untrusted code and provides students with real-time, automated feedback. Unlike conventional Online Judge platforms designed primarily for competitive programming, this system is specifically tailored to the pedagogical needs of novice learners. The resulting platform offers a robust, secure, and pedagogically enhanced solution for online programming assessment, improving both the efficiency for instructors and the learning experience for students.

Keywords: Online Judge, Programming Education, Interactive Learning Environment, Java, Real-time Feedback.

ISSN 2407-4489
pp. 113-121

INTRODUCTION

Education in the digital era, often referred to as Education 4.0, demands significant improvements in the quality, speed, and practicality of the learning process. One aspect undergoing a profound transformation is the evaluation process, particularly in technical disciplines such as computer programming. Traditional assessment methods, which rely on manual correction by instructors, often face numerous challenges. This process is not only time-consuming but also susceptible to subjectivity and inconsistency, especially when handling complex submissions like program code or specific syntax (Mustamiin et al., 2023). Delays in providing feedback also pose a crucial problem, as students lose the opportunity to promptly identify and correct their errors, which can ultimately impede their learning progress (Mustamiin et al., 2023).

In response to these challenges, web-based online examination systems have become an increasingly popular and essential solution (Saputra & Mulyoto, 2023). According to research by Stefany et al. (2024), developing website-based e-learning media has been shown to be an attractive and flexible alternative for student teaching materials. This development is driven by the need to create a more efficient, objective, and measurable evaluation process. Among the various online examination technologies, the Online Judge (OJ) system stands out as a platform specifically designed for the automated and reliable evaluation of program code (Wasik et al., 2018). Initially developed for competitive programming contests like the ACM ICPC, the application of OJs has now expanded into formal educational contexts (Wasik et al., 2018). An OJ system can provide in-depth evaluations that not only check the correctness of the program's output but also analyze other aspects such as execution time efficiency, memory usage, and compilation errors (Mustamiin et al., 2023). However, a significant gap exists in current technology: a modern assessment platform should not function merely as a testing tool. To be effective, it must evolve into an Interactive Learning Environment (ILE). The ILE concept emphasizes the importance of interaction between students and learning materials to enhance learning outcomes and engagement (Villardón-Gallego et al., 2018). Various studies have shown that incorporating an Interactive Learning Platform (ILP) into programming courses significantly improves student performance. An interactive environment helps students overcome the fear and difficulties in learning programming logic and fosters essential computational thinking skills (Grover & Pea, 2013). Thus, the assessment paradigm shifts from merely being an assessment of learning to assessment as learning.

Based on this background, this research identifies the need to develop an online examination platform that not only efficiently and securely evaluates Java programming code but also functions as an ILE that actively supports the student learning process. Therefore, the objective of this research is to design and build a Java programming examination website that integrates Online Judge functionality for automated assessment with an interactive interface to provide real-time feedback.

METHOD

The methodology utilized in the development of this system encompasses a structured software development model, the design of a modular system architecture, and the selection of appropriate technologies to support both functional and pedagogical objectives.

2.1 System Development Model

The development of this system adopts the waterfall model to ensure systematic alignment between pedagogical requirements and technical implementation. This model was chosen due to its sequential and structured nature, which is highly suitable for projects with requirements that are clearly defined at the outset of development (Royce, 1970). This approach ensures that each development phase—from requirements analysis, design, implementation (coding), to testing—is conducted

ISSN 2407-4489
pp. 113-121

systematically and is well-documented, thereby minimizing potential errors and resulting in a higher-quality software product. The stages undertaken in the waterfall model for this research are as follows:

a. Requirements Analysis: Identifying the functional and non-functional requirements of the system from the perspectives of both students and instructors.
b. System Design: Designing the system architecture, database schema, and user interface/user experience (UI/UX) design.
c. Implementation: Translating the design into program code using the specified technologies.
d. Testing: Conducting functional testing to ensure all features operate as expected and are free of errors.

2.2 System Architecture

This system is designed with a decoupled architecture, inspired by the principles of microservices (Newman, 2015). This architecture separates the system into three main components that interact with each other through an Application Programming Interface (API), as illustrated in Figure 1.

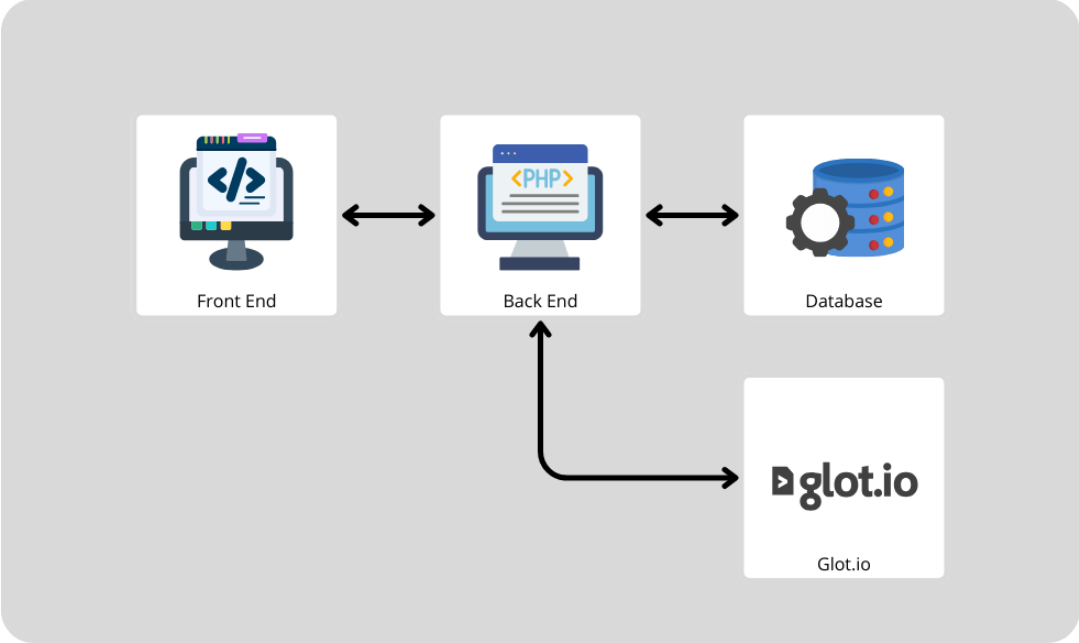


Fig 1. Online Examination System Architecture

The components of this architecture are as follows:

a. Front-End Client (Client Application): This is the web-based interface accessed by users (students and instructors) through a browser. This component is responsible for presenting the display, handling user interactions, and communicating with the back-end server. The central feature of this layer is an interactive code editor.
b. Back-End Server (Server Application): This acts as the brain of the system. This component handles all business logic, including user authentication and management, question and exam management, storage of code submission data, and the orchestration of communication between the front-end and the external code execution service.
c. External Code Execution Service: For the code evaluation function, the system delegates the execution task to an external service, namely the Glot.io API. This architectural decision is crucial. Building a secure execution environment (sandbox) from scratch is a highly complex and high-risk task. Malicious code submitted by a user could cause Denial-of-Service (DoS) attacks, access unauthorized files, or even compromise the server environment (Liang et al.,

2017; Wu et al., 2022). By using a proven external API like Glot.io, which runs each code submission in an isolated and temporary Docker container, these security risks are offloaded to the service provider. This approach, known as Evaluation-as-a-Service (EaaS) (Wasik et al., 2018), allows the development team to focus more on creating pedagogical features and a user interface that supports the learning process.

2.3 Technologies Used

The technology selection is based on the functional requirements of each component in the architecture.

a. Code Editor (Front-End): To create an interactive learning environment, CodeMirror is utilized, a JavaScript library that provides a code editor component for the web (Haverbeke, 2024). CodeMirror was chosen for its rich feature set that supports the programming learning process, such as syntax highlighting, line numbering, bracket matching, and autocompletion. The syntax highlighting feature has been proven to significantly improve code readability, help programmers find errors more quickly, and reduce cognitive load, which allows students to focus more on understanding the program's logic (Sarkar, 2015).
b. Code Execution (Back-End Integration): For code execution and evaluation, the system integrates with the Glot.io API. The code evaluation process is designed as follows:
    1) The front-end sends the Java source code written by the student, along with the problem ID, to a back-end script (PHP).
    2) Simultaneously, the back-end script performs two tasks: it retrieves the expected output from the database based on the problem ID, and forwards the student's source code to the Glot.io API for execution.
    3) Glot.io runs the code in a secure and isolated environment, then returns the actual output to the back-end server.
    4) The back-end server compares the actual output from Glot.io with the expected output from the database.
    5) The comparison result (a status of Correct or Incorrect), along with the actual output, is sent back to the front-end in JSON format to be displayed to the user.

RESULTS AND DISCUSSION

This section presents the implementation results of the developed system, followed by an in-depth discussion on the significance of its features from both technical and pedagogical perspectives, linking them to relevant existing literature.

3.1 Results: System Feature Implementation

The successfully developed online examination system has functionalities divided based on two main user roles: the Instructor and the Student.

3.1.1 Features for the Instructor:

a. Exam Management: Instructors can create new exam sessions, set their duration, and determine the questions to be included.

ISSN 2407-4489
pp. 113-121

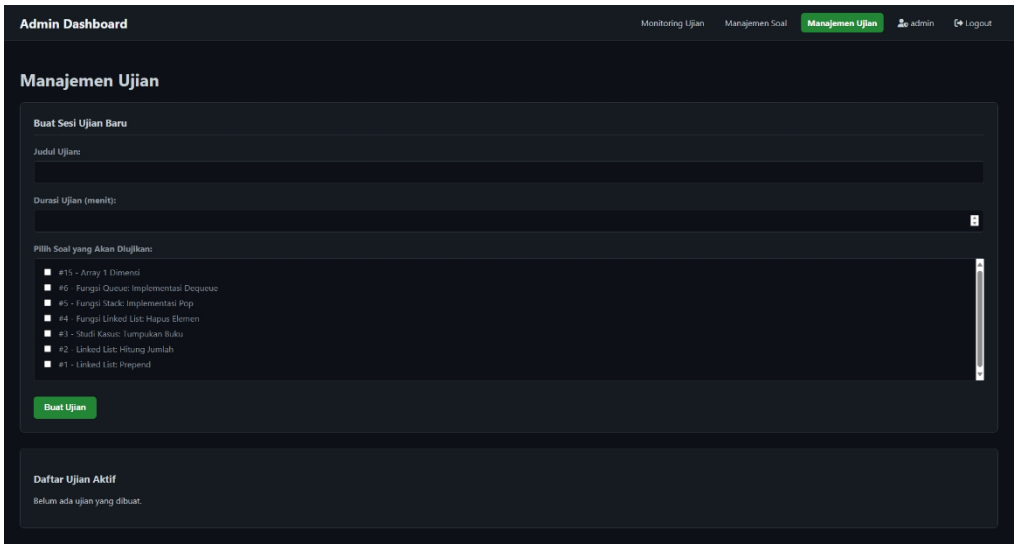


Fig 2. Exam Management Dashboard

b. Problem Management: Instructors can create, modify, and delete the problem bank. Each problem consists of a description, sample inputs/outputs, and hidden test cases with their expected outputs.

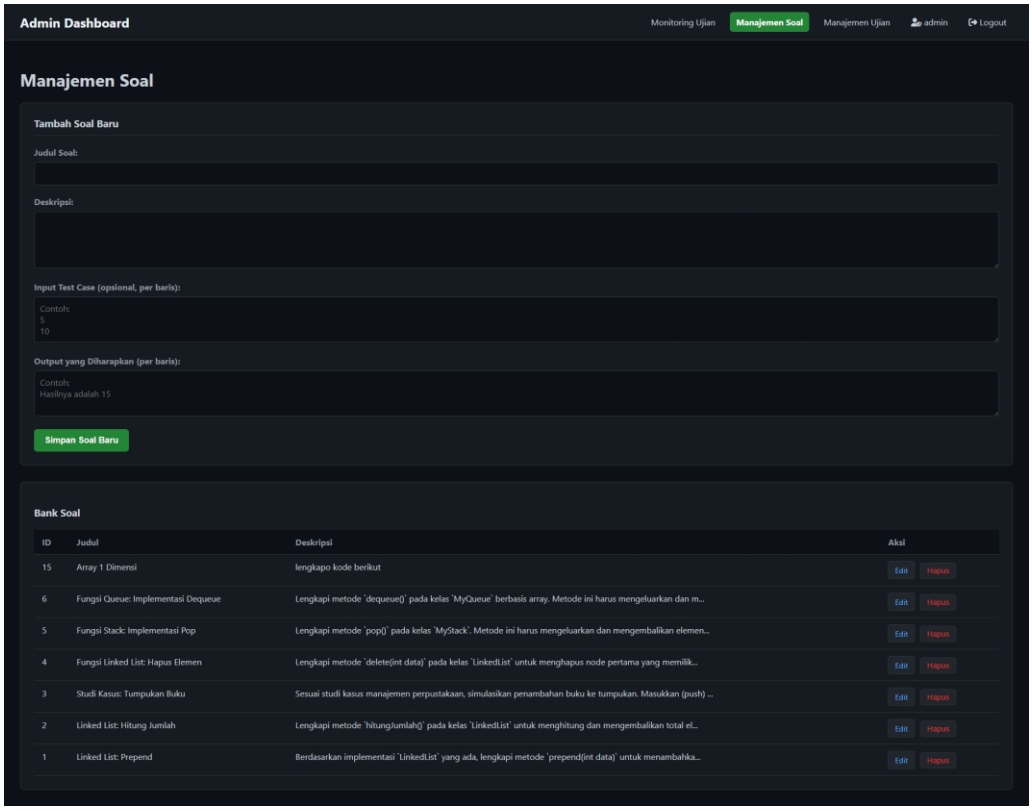


Fig 3. Problem Management Dashboard

c. Real-time Monitoring: Instructors can monitor exam progress live, view the list of students currently working, and see the status of each student's code submissions. This feature allows instructors to get a general overview of the problem difficulty and the overall class performance.

ISSN 2407-4489
pp. 113-121

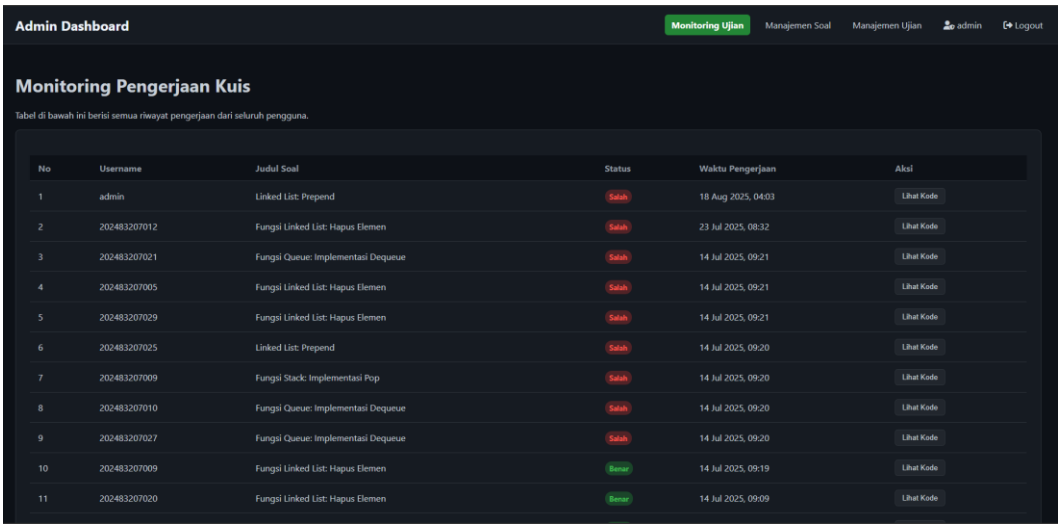


Fig 4. Programming Quiz Monitoring Dashboard

3.1.2 Features for the Student:

a. Examination Interface: After joining an exam session, students are presented with a clean and focused interface. As shown in Figure 2, this interface displays the problem description on one side and an interactive code editor on the other.
b. Interactive Code Editor: Utilizing CodeMirror, the editor provides a comfortable coding environment with features like syntax highlighting for Java, line numbering, and automatic indentation.
c. Code Submission and Instant Feedback: Students can write their solutions in the editor and submit them for evaluation at any time during the exam. Within seconds, the system displays the result of the automated evaluation, such as "Correct Answer" or "Incorrect Answer," based on the comparison of their program's output with the answer key. This instant feedback is a key advantage of OJ systems (Mustamiin et al., 2023).

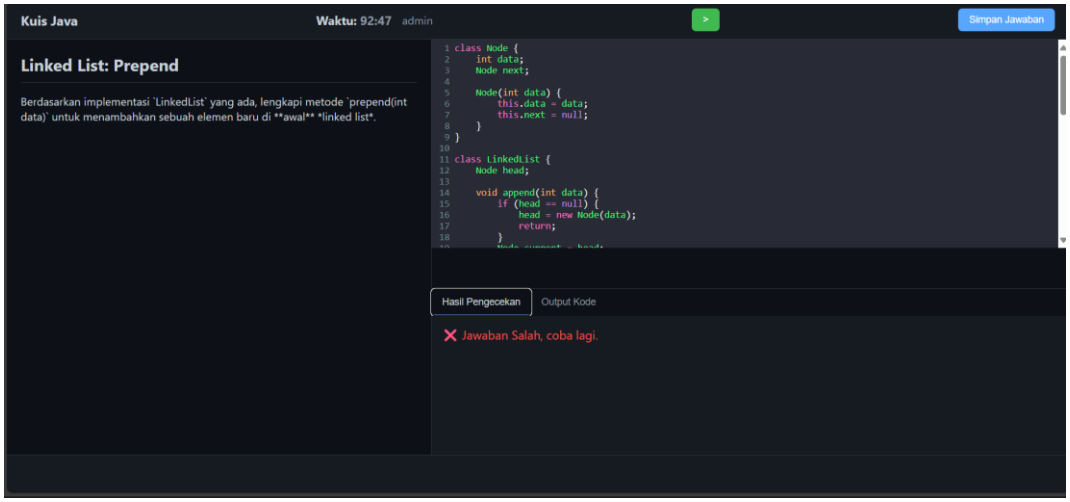


Fig 5. Student Examination Interface

To summarize the main contributions of this system, Table 1 presents a mapping between the implemented technical features and their underlying pedagogical rationales.

ISSN 2407-4489
pp. 113-121

Table 1. Key Features and Their Pedagogical Rationales

Feature	Technology Implementation	Pedagogical Rationale & References
Interactive Code Editor	CodeMirror	Reduces cognitive load and improves code readability, allowing students to focus on programming logic (Sarkar, 2015).
Automated Code Evaluation	Glot.io API Integration	Provides objective, consistent, and instant evaluation, eliminating the bias and delays of manual grading (Mustamiin et al., 2023; Wasik et al., 2018).
Real-time Feedback	Comparison of Glot.io API output with database data	Facilitates a rapid self-correction learning cycle, turning the exam into a formative learning experience (Mustamiin et al., 2023).
Secure Execution Environment	Isolated code execution via Glot.io API	Protects system integrity from malicious code and ensures a fair and reliable examination environment for all participants (Liang et al., 2017; Wu et al., 2022).

3.2 Discussion

The implementation of this system carries several important implications that warrant further discussion. This discussion focuses on how the chosen architecture and features synergistically create a platform that is not only technically superior but also pedagogically robust.

3.2.1 Creating an Interactive Learning Environment (ILE)

The integration of CodeMirror is more than just an aesthetic enhancement. Compared to a plain textarea, this interactive editor actively supports the student's cognitive processes. Features like syntax highlighting act as "visual cues" that help students distinguish between keywords, variables, strings, and comments at a glance (Sarkar, 2015). This reduces extraneous cognitive load—the mental effort spent on understanding syntax—allowing the student's cognitive resources to be fully allocated to the more critical tasks: problem-solving and building program logic (Sweller, 1994). Thus, the system actively contributes to the creation of an ILE that supports the learning process, rather than merely testing the final output.

3.2.2 Benefits of Real-time Feedback for Formative Learning

One of the most significant pedagogical transformations offered by this system is the shift from summative to formative assessment. In the traditional model, feedback is given long after an exam is over, limiting its function to a mere justification of a grade (*assessment of learning*). In contrast, the instant feedback provided by the Glot.io integration enables a rapid learning cycle: try, fail, analyze, correct, and try again. Each time a student submits code and receives an "Incorrect Answer" response, they are prompted to reflect on their mistakes and try a different approach. This iterative process transforms the exam into an intense and focused practice session, where learning actively occurs within the assessment environment itself (*assessment as learning*) (Mustamiin et al., 2023). For instructors, this real-time data provides valuable insights into which concepts students find most difficult, allowing for more timely and effective teaching interventions.

3.2.3 Architectural Advantages in Security and Scalability

The architectural decision to delegate code execution to an external service like Glot.io offers a dual advantage: security and scalability. On the security front, the core system is protected from the threat of malicious code, ensuring the platform's integrity and availability during an exam (Liang et al., 2017; Wu et al., 2022). This is achieved because Glot.io runs every submitted piece of code in an isolated, temporary Docker container, which is destroyed after execution is complete. This is a crucial non-functional aspect for any assessment platform. In terms of scalability, because the resource-intensive execution process occurs on an external server, the main application's back-end can handle

more concurrent users more efficiently. If the execution load increases, scalability can be achieved by adding resources to the external service without needing to alter the core application's architecture.

3.2.4 The System's Position Among Other Platforms

When compared to existing OJ platforms, this developed system occupies a specific niche. Many popular platforms like UVa, DMOJ, or Kattis are designed primarily for the competitive programming community (Wasik et al., 2018). While powerful, these platforms often have a steep learning curve and features that may be less suited for the context of a structured university classroom. This system, in contrast, is designed with the specific needs of an educational environment in mind: a user-friendly interface, an intuitive workflow for instructors to create and manage exams, and a focus on feedback as a learning tool. Thus, the system balances the rigor of technical OJ-style evaluation with the pedagogical support required in formal education.

CONCLUSIONS

The development of this Java programming examination website has successfully demonstrated that the integration of an interactive user interface with a secure, external code execution engine can yield an effective and reliable assessment platform. The developed system is capable of overcoming the main challenges in programming assessment such as subjectivity, inefficiency, and delayed feedback—by providing objective, consistent, and instant automated evaluation. The implementation of a decoupled architecture proved to be a superior strategic approach, as it allows developers to focus on refining pedagogical features and the user experience while delegating the complex and high-risk code execution function to a specialized external service. The result is an interactive learning environment that not only tests students' abilities but also actively supports their learning process through a real-time feedback cycle.

Limitations of the Research: Despite the successful development, this study has limitations. Currently, the system has only been tested for functional reliability and has not yet undergone large-scale empirical testing in a real classroom setting to measure its direct impact on student grades. Additionally, the current problem bank is limited to basic Java syntax.

Future Work: Future work will focus on deploying this system in an introductory programming course to gather longitudinal data on student performance and user experience (UX). Scalability testing with concurrent users will also be conducted. For future development, the system can be enhanced by adding a plagiarism detection module to further ensure academic integrity (Islam et al., 2021), as well as integrating artificial intelligence (AI) to provide more contextual and personalized feedback, moving towards the realization of a more comprehensive Automated Programming Assessment System (APAS). This aligns with the findings of Putra et al. (2025), who reported that students found AI to be very useful for enhancing their understanding in learning.

.

ACKNOWLEDGMENT

The author wishes to express sincere gratitude to all parties who have provided valuable support and contributions towards the completion of this research and the writing of this article. Thanks are extended to the Mojosari Institute of Technology for providing a supportive academic environment and the necessary facilities during the research process.

REFERENCES

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38–43. https://doi.org/10.3102/0013189X12463051

Haverbeke, M. (2024). CodeMirror (Version 5.65) [Computer software]. https://codemirror.net/

Islam, M. R., Reza, M. T., Noman, S. M., Islam, M. R., & Islam, M. S. (2021). Online judging platform utilizing dynamic plagiarism detection facilities. Computers, 10(4), 47. https://doi.org/10.3390/computers10040047

Liang, H., Chen, C., Zhong, X., & Chen, Y. (2017). Design and implementation of online automatic judging system. IOP Conference Series: Earth and Environmental Science, 69(1), 012091. https://doi.org/10.1088/1755-1315/69/1/012091

Mustamiin, M., Suheryadi, A., & Puspaningrum, A. (2023). Penerapan sistem evaluasi pembelajaran pemrograman terintegrasi dengan online judge di SMK. Jurnal Pengabdian Kepada Masyarakat, 2(2), 61–67. https://doi.org/10.46961/jpk.v2i2.636

Newman, S. (2015). Building microservices: Designing fine-grained systems. O'Reilly Media.

Putra, M. T. M., Rochmah, I. N., Sa'adah, N. L., Sabar., & Sari, N. P. (2025). Analisis Penerimaan Teknologi Kecerdasan Buatan dalam Pembelajaran Pemrograman Web: Pendekatan Model Penerimaan Teknologi. Jurnal Ilmiah Edutic: Pendidikan dan Informatika, 12(1), 22-28. https://doi.org/10.21107/edutic.v12i1.28877

Royce, W. W. (1970). Managing the development of large software systems. Proceedings of IEEE WESCON, 1–9.

Saputra, E. A., & Mulyoto, A. (2023). Perancangan sistem ujian online berbasis website pada SDN Rawa Mekar Jaya. Jurnal Teknologi Sistem Informasi dan Aplikasi, 6(1), 1-8.

Sarkar, A. (2015). The impact of syntax highlighting on program comprehension. Proceedings of the 26th Annual Workshop of the Psychology of Programming Interest Group, 5.

Stefany, E. M., Dellia, P., & Khoiroh, M. D. (2024). Pengembangan E-Learning Sistem Komputer Kelas X di SMKN 1 Kwanyar. Jurnal Ilmiah Edutic: Pendidikan dan Informatika, 11(1), 65–79. https://doi.org/10.21107/edutic.v11i1.28151

Sweller, J. (1994). Cognitive load theory, learning difficulty, and instructional design. Learning and Instruction, 4(4), 295–312. https://doi.org/10.1016/0959-4752(94)90003-5

Villardón-Gallego, L., García-Carrión, R., Yáñez-Marquina, L., & Estévez, A. (2018). The impact of interactive learning environments on the development of prosocial behavior in elementary school students. Frontiers in Psychology, 9, 2323. https://doi.org/10.3389/fpsyg.2018.02323

Wasik, S., Antczak, M., Badura, J., Laskowski, A., & Sternal, T. (2018). A survey on online judge systems and their applications. ACM Computing Surveys, 51(1), Article 3. https://doi.org/10.1145/3143560

Wu, Z. J., Kuo, J. Y., & Huang, H. X. (2022). Security analysis of online judge systems. Applied Sciences, 12(14), 7318. https://doi.org/10.3390/app12147318