Web-Based Programming Assessment with DeepSeek R1 AI Feedback System

Windra Swastika[1,*], Vincent Dwi Hartanto[1], and Paulus Lucky Tirma Irawan[1]

[1]Informatics Engineering, Faculty of Technology and Design, Universitas Ma Chung, Malang, Indonesia

Corresponding author: Windra Swastika (e-mail: windra.swastika@machung.ac.id).

ABSTRACT The evaluation of programming assignments in computer science education is often time-consuming and subjective when done manually by instructors. This study proposes an automated web-based code evaluation system that leverages Artificial Intelligence (AI) to provide immediate and objective feedback. The system is built with the Next.js framework for the user interface and integrates a Large Language Model (LLM), DeepSeek-R1, via the LangChain framework as the intermediary for AI communication. Through this integration, the system automatically analyses student source code submissions against given programming tasks and grading rubrics, generating feedback and scores in real-time. Performance testing of the implemented system yielded an average response time of 5.7 seconds, a throughput of 10.86 requests per second, and a front-end load performance score of 78. User evaluation using the System Usability Scale (SUS) resulted in a score of 77, indicating a positive user experience. The system functioned effectively during trials, demonstrating that AI-driven code evaluation can accelerate and simplify the grading process in programming education while maintaining satisfactory accuracy and usability. These findings suggest that integrating LLMs into programming education platforms offers a scalable and effective approach to automated assessment, with significant potential to enhance learning outcomes and reduce the workload of instructors in academic environments.

KEYWORDS Artificial Intelligence, Code Evaluation, Deepseek-R1, Large Language Model

I. INTRODUCTION

The algorithm course is one of the fundamental pillars in the curriculum of Informatics study programs. An algorithm is broadly defined as a finite sequence of well-defined, computer-implementable instructions to solve a class of problems [1]. This course is crucial as it equips students with logical thinking skills, systematic algorithm design, and implementation in programming languages. As emphasized by various educators, computer programming is a skill that must be mastered by computer science students, making the learning and evaluation processes in this course essential for ensuring students truly understand the taught material [2].

However, in practice, the evaluation and assessment of students' program code still face various problems. Providing feedback on programming assignments is a very tedious, error-prone, and time-consuming task for instructors, even in standard classroom environments [3]. Additionally, manual assessment is often subjective and inconsistent, which poses a significant challenge in computer science education [4]. Students often only receive final grades without clear and detailed feedback regarding deficiencies or errors in their program code, which occurs because the feedback provided is ineffective, thus not beneficial for students and cannot improve their performance [5]. The lack of immediate, constructive feedback is a significant barrier to student learning and improvement in programming courses.

With technological development, Artificial Intelligence (AI) has presented innovative solutions in various fields, including education. Dewi [6] defines artificial intelligence as a field of study related to capturing, modeling, and storing human intelligence in an information technology system so that the system can be used as a decision-making process performed by humans. Zulkarnain and Yunus [7] emphasized that the use of AI technology in learning can serve as a powerful computational solution, where AI-powered technology can be used to assist teaching and evaluation processes. A systematic review of AI applications in higher education further confirmed this trend, finding that AI is increasingly being applied across various educational domains including assessment, intelligent tutoring, and personalized learning [8].

One significant development in AI is the emergence of Large Language Models (LLMs), which can understand and generate text in natural language as well as programming languages. LLMs are large-scale language models trained on vast datasets, giving them extensive language understanding and knowledge without needing highly structured data to process and understand specific natural languages [9]. Models like DeepSeek-R1 are examples of LLMs specifically designed to understand, evaluate, and provide automatic feedback on program code. Recent studies have shown promising results regarding the effectiveness of DeepSeek models in programming tasks. According to the model's developers, DeepSeek-R1 achieved high scores on mathematical and coding benchmarks, indicating that this model can outperform a significant percentage of human participants in programming competitions [10]. Furthermore, comparative studies have demonstrated that DeepSeek shows competitive performance against other Large Language Models while offering significant advantages in terms of cost efficiency [11]. One study found that DeepSeek outperformed ChatGPT in solving programming contest problems, with a correctness rate of 0.5454 compared to ChatGPT's 0.1875 [12].

By utilizing AI capabilities in data analysis and modelling, many new approaches have been proposed to measure student performance more accurately and efficiently [13]. LLMs like DeepSeek-R1 can be integrated with LangChain, a framework that enables the development of AI-based applications in a modular and flexible manner. LangChain is an open-source framework that facilitates the creation of LLM applications with custom data, high scalability, and accelerated development processes [14]. Therefore, a web-based program code evaluation platform was developed using the Next.js framework. This platform integrates LangChain to connect the user interface with the DeepSeek-R1 LLM, enabling automatic and interactive program code evaluation processes. With this system, instructors are expected to evaluate more easily while students receive beneficial feedback to improve their programming skills effectively and efficiently.

Despite the growing body of research on AI-assisted programming assessment, existing solutions present several limitations that our system addresses. Traditional auto graders such as CodeRunner and AutoGrader rely primarily on static test cases and predefined rules, providing limited explanatory feedback beyond pass/fail indicators [18]. While recent tools like GitHub Copilot for Education and ChatGPT-based assessment systems leverage large language models for code analysis, they often lack integration with learning management workflows and fail to provide structured, rubric-aligned evaluations [19]. Our system introduces three key innovations that distinguish it from existing approaches. First, we implement real-time, rubric-guided AI feedback that generates detailed explanations aligned with specific learning objectives, moving beyond binary correctness to provide educational insights. Second, our architecture integrates LangChain as a modular intermediary layer, enabling flexible LLM selection and prompt engineering strategies that can be adapted to different programming languages and assessment contexts. Third, we provide comprehensive scalability evidence through systematic load testing and economic analysis, demonstrating the practical viability of deploying AI-powered assessment at institutional scale. The use of DeepSeek-R1 specifically offers advantages in cost-efficiency and reasoning capabilities for programming tasks, as demonstrated by its competitive performance on coding benchmarks [10,12]. Our contribution lies not merely in applying an LLM to programming assessment, but in creating an end-to-end educational platform that balances pedagogical effectiveness, technical performance, and economic sustainability.

II. RESEARCH METHOD

This research uses a software engineering methodology aimed at designing and building a web-based system for automating source code evaluation using artificial intelligence (AI). The research was conducted with a structured approach through several interrelated stages as illustrated in Figure 1.

A. Research Design

The research adopts a development research methodology, which combines theoretical foundations with practical implementation to create and validate educational products [15]. The study follows a sequential approach beginning with literature review, progressing through system requirements analysis, design, implementation, testing, and evaluation. Figure 1 illustrates the research methodology flowchart that guides the entire development process from initial conceptualization to final documentation.

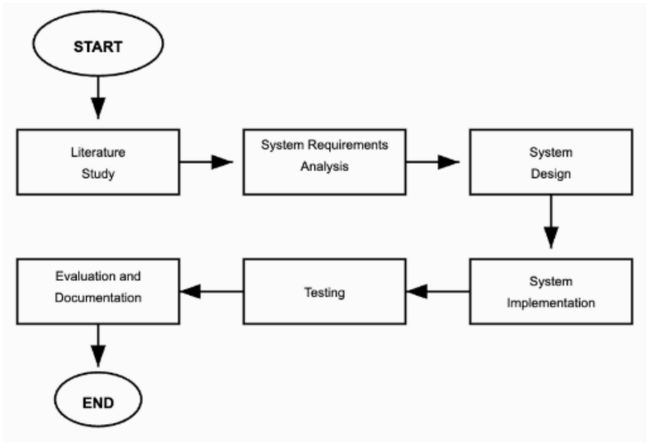


Figure 1. Research Flow Diagram

The development process initiates with literature study to establish theoretical foundations and identify existing gaps in automated programming assessment systems. This phase involves comprehensive analysis of current AI-based evaluation techniques, learning management systems, and programming education methodologies. The literature review provides insights into state-of-the-art approaches and establishes the research novelty by identifying limitations in existing solutions.

Following the literature study, the system requirements analysis phase determines functional and non-functional requirements for the platform. This analysis encompasses user requirements for both students and educators, technical specifications for AI integration, performance criteria, and security considerations. The requirements analysis ensures that the developed system addresses real educational needs while maintaining technical feasibility and scalability.

B. System Design and Architecture

The system design phase establishes the overall architecture and detailed specifications for the AI-powered assessment platform. Figure 2 presents the activity diagram that illustrates the interaction between three primary actors: Students, System, and Educators. The architecture shows the complete workflow from user registration and authentication through AI-based evaluation and feedback delivery.

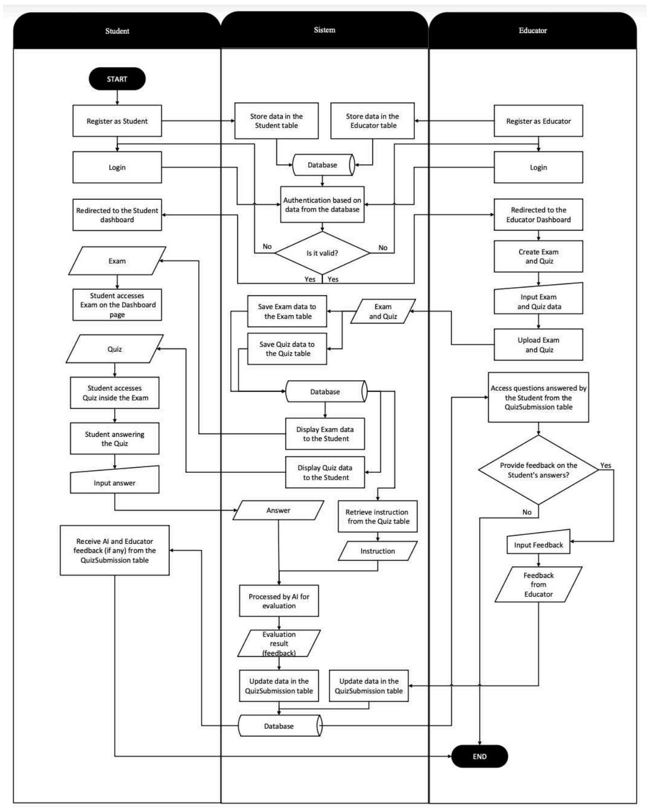

Student
Sistem
Educator
START
Register as Student
Store data in the Student table
Store data in the Educator table
Register as Educator
Login
Database
Login
Authentication based on data from the database
Redirected to the Student dashboard
Redirected to the Educator Dashboard
No
Is it valid?
No
Yes
Yes
Exam
Create Exam and Quiz
Student accesses Exam on the Dashboard page
Save Exam data to the Exam table
Exam and Quiz
Input Exam and Quiz data
Save Quiz data to the Quiz table
Upload Exam and Quiz
Quiz
Student accesses Quiz inside the Exam
Database
Access questions answered by the Student from the QuizSubmission table
Student answering the Quiz
Display Exam data to the Student
Display Quiz data to the Student
Provide feedback on the Student's answers?
Yes
Input answer
Answer
Retrieve instruction from the Quiz table
No
Receive AI and Educator feedback (if any) from the QuizSubmission table
Instruction
Input Feedback
Feedback from Educator
Processed by AI for evaluation
Evaluation result (feedback)
Update data in the QuizSubmission table
Update data in the QuizSubmission table
Database
END

Figure 2. Activity Diagram

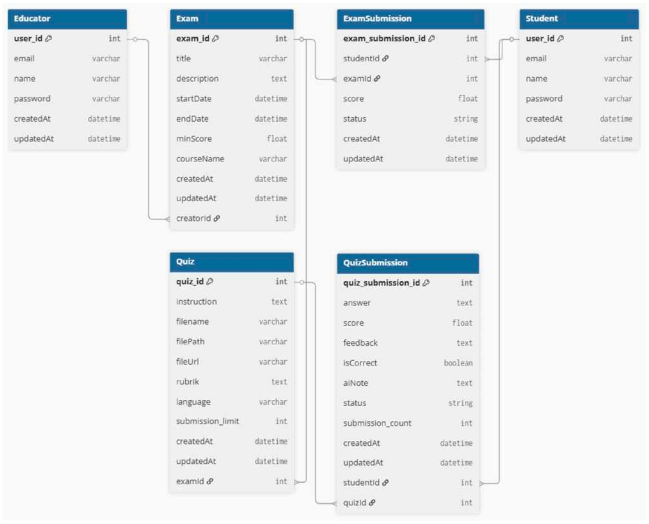
The system incorporates a web-based platform that facilitates seamless interaction between students and educators while integrating artificial intelligence for automated assessment. The design emphasizes modular architecture to ensure maintainability, scalability, and extensibility. The platform utilizes LangChain as an intermediary framework to manage interactions with the DeepSeek-R1 large language model, enabling sophisticated natural language processing capabilities for code evaluation and feedback generation.

The functional requirements include user authentication for both students and mentors, exam and assignment upload capabilities by mentors, student answer submission through web-based code editor, automatic AI evaluation using DeepSeek-R1, automatic feedback provision, mentor access to student assignments and AI evaluation results, optional additional mentor feedback, and comprehensive data storage to PostgreSQL database through Prisma ORM.

Non-functional requirements encompass system responsiveness and accessibility across various devices, optimal system performance with reasonable AI evaluation processing time, high system availability (24/7 uptime), and user-friendly interface design that is easily understood by both students and mentors.

C. Database Design

The database design employs a relational database model to manage system data efficiently. Figure 3 illustrates the Entity Relationship Diagram (ERD) that defines the relationships between core entities including Educator, Student, Exam, Quiz, ExamSubmission, and QuizSubmission. The database schema ensures data integrity through appropriate constraints, foreign key relationships, and normalization principles.



Educator	
user_id	int
email	varchar
name	varchar
password	varchar
createdAt	datetime
updatedAt	datetime

Exam	
exam_id	int
title	varchar
description	text
startDate	datetime
endDate	datetime
minScore	float
courseName	varchar
createdAt	datetime
updatedAt	datetime
creatorId	int

ExamSubmission	
exam_submission_id	int
studentId	int
examId	int
score	float
status	string
createdAt	datetime
updatedAt	datetime

Student	
user_id	int
email	varchar
name	varchar
password	varchar
createdAt	datetime
updatedAt	datetime

Quiz	
quiz_id	int
instruction	text
filename	varchar
filePath	varchar
fileUrl	varchar
rubrik	text
language	varchar
submission_limit	int
createdAt	datetime
updatedAt	datetime
examId	int

QuizSubmission	
quiz_submission_id	int
answer	text
score	float
feedback	text
isCorrect	boolean
aiNote	text
status	string
submission_count	int
createdAt	datetime
updatedAt	datetime
studentId	int
quizId	int

Figure 3. Entity relationship diagram

The Educator entity stores instructor information including authentication credentials and profile data. The Student entity maintains learner information with similar authentication and profile attributes. The Exam entity represents assessment containers created by educators, while the Quiz entity defines individual programming problems within examinations. The ExamSubmission entity tracks student participation in examinations, and the QuizSubmission entity records detailed submission data including source code, AI-generated feedback, scores, and evaluation status.

D. Implementation Methodology

The system implementation follows an iterative development approach utilizing modern web technologies and frameworks. The platform development incorporates responsive web design principles to ensure accessibility across various devices and screen sizes. The implementation integrates AI capabilities through API connections to language models, enabling real-time code evaluation and feedback generation.

The development process emphasizes security considerations including user authentication, comprehensive error handling, logging mechanisms, and performance optimization techniques. The modular architecture facilitates independent development and testing of system components while maintaining overall system cohesion.

E. Testing and Evaluation

The testing methodology encompasses multiple phases including unit, integration, system, and user acceptance testing. The evaluation phase includes performance metrics assessment, AI evaluation accuracy measurement, and user experience analysis using the System Usability Scale (SUS) [16]. The SUS is a widely adopted and reliable tool for measuring perceived usability through a ten-item questionnaire.

The evaluation phase includes performance metrics assessment, AI evaluation accuracy measurement, and user experience analysis. The testing process validates system requirements compliance and identifies areas for improvement. User feedback collection through surveys and usage analytics provides insights into system effectiveness and user satisfaction levels.

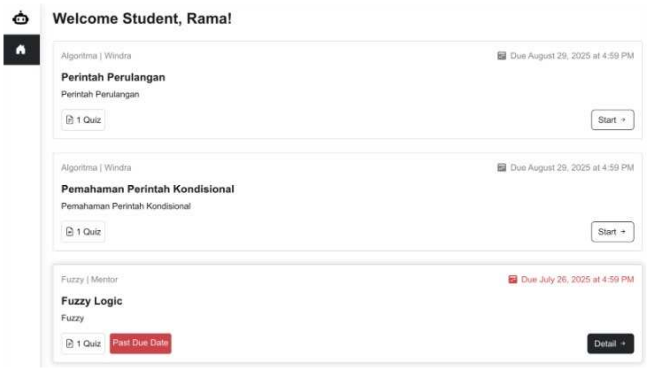
The research methodology ensures systematic development of a comprehensive AI-powered assessment platform that addresses current limitations in programming education while providing innovative solutions for automated code evaluation and personalized feedback delivery.

III. *RESULTS AND DISCUSSION*

A. System Implementation Results

The developed system successfully integrates all planned components into a functional web-based platform. The system architecture utilizes Next.js for both frontend and backend development, with PostgreSQL database managed through Prisma ORM, and DeepSeek-R1 AI model accessed via official API through LangChain framework.

Several implementation adjustments were made from the original design: the student answer collection method was changed from file upload to direct code editor input for better user experience and real-time processing capability. The user interface implementation closely follows the initial wireframe designs, with minor adjustments for improved usability. Figure 4 shows the main dashboard interface for students, displaying available exams with relevant information including course name, instructor, deadline, and completion status.



Welcome Student, Rama!

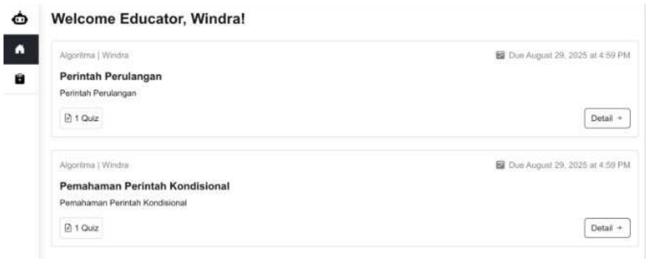
Algoritma | Windra — Due August 29, 2025 at 4:59 PM
Perintah Perulangan
Perintah Perulangan
1 Quiz — Start

Algoritma | Windra — Due August 29, 2025 at 4:59 PM
Pemahaman Perintah Kondisional
Pemahaman Perintah Kondisional
1 Quiz — Start

Fuzzy | Mentor — Due July 26, 2025 at 4:59 PM
Fuzzy Logic
Fuzzy
1 Quiz — Past Due Date — Detail

Figure 4. **Student dashboard interface**

The mentor interface provides comprehensive access to student submissions and AI evaluation results, as demonstrated in Figure 5. This interface allows mentors to review AI-generated feedback and provide additional manual feedback when necessary.



Welcome Educator, Windra!

Algoritma | Windra — Due August 29, 2025 at 4:59 PM
Perintah Perulangan
Perintah Perulangan
1 Quiz — Detail

Algoritma | Windra — Due August 29, 2025 at 4:59 PM
Pemahaman Perintah Kondisional
Pemahaman Perintah Kondisional
1 Quiz — Detail

Figure 5. **Mentor evaluation interface**

B. AI Evaluation Implementation

The AI evaluation system processes student submissions through a prompt engineering approach. The system constructs prompts using markdown format for better structure recognition by the AI model. Each evaluation request includes the programming language specification, problem statement, evaluation rubric, and student's code submission.

The evaluation process generates three key outputs: detailed narrative feedback explaining the analysis and suggestions for improvement, quantitative score based on the provided rubric, and binary correctness status (true/false). Figure 6 illustrates the complete AI evaluation workflow from submission to result display.

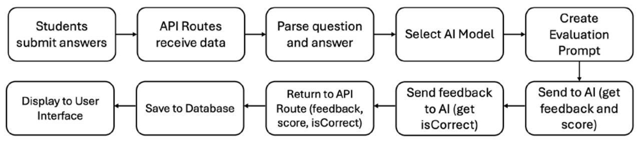

Students submit answers
API Routes receive data
Parse question and answer
Select AI Model
Create Evaluation Prompt
Send to AI (get feedback and score)
Send feedback to AI (get isCorrect)
Return to API Route (feedback, score, isCorrect)
Save to Database
Display to User Interface

Figure 6. **AI evaluation workflow**

Sample evaluations demonstrate the system's capability to provide constructive feedback. For a correct solution to a temperature conversion problem, the AI provided detailed analysis highlighting the proper implementation while suggesting minor improvements for edge case handling. For an incorrect solution with logical errors, the AI identified specific issues and provided guidance for correction without directly revealing the solution.

C. Performance Testing Results

Performance testing was conducted to evaluate system capabilities under various load conditions. The results are summarized in Table 1, showing key performance metrics obtained through different testing tools.

TABLE I
PERFORMANCE TESTING RESULTS

Metric	Value	Tool Used
Average Response Time	5.7 sec	Grafana K6
Minimum Response Time	3.2 sec	Grafana K6
Maximum Response Time	23.7 sec	Grafana K6
Throughput	10.8 req/sec	Grafana K6
Load time performance score	78	Google Lighthouse
System Usability Scale (SUS)	77	User survey

The response time analysis shown in Table I indicates that while the average processing time of 5.7 seconds is reasonable for AI-powered evaluation, there are occasional longer processing times that may require optimization for improved user experience.

D. System Usability Evaluation

The System Usability Scale (SUS) evaluation involved five student participants. The average SUS score of 77 indicates good system usability. According to established benchmarks, a SUS score above 68 is considered above average, and a score of 77 falls into the "Good" or "B grade" category, confirming a positive user reception [17]. Qualitative feedback from participants revealed positive aspects including the clarity of AI feedback and the fairness of the partial credit scoring system.

The average SUS score of 77 indicates good system usability, as scores above 68 are generally considered above-average globally. This result suggests that the system provides a satisfactory user experience for students in programming course contexts.

Qualitative feedback from participants revealed positive aspects including clear and comprehensive AI feedback, helpful explanations for error correction, and fair scoring system that provides partial credit rather than binary pass/fail grades. Areas for improvement identified include interface design enhancement, addition of code editor features like auto-completion, and faster AI processing times.

E. AI Evaluation Accuracy and Human Agreement Analysis

To validate the accuracy of AI-generated assessments, we conducted a systematic comparison between DeepSeek-R1 evaluations and expert human grading. A dataset of 20 student submissions across three programming problems (basic algorithms and medium-difficulty problems) was independently evaluated by one experienced programming instructors and the AI system. Table II presents the agreement analysis between AI and human evaluations. The scoring comparison shows strong correlation between AI and human assessments, with a Pearson correlation coefficient of 0.847 ($p < 0.001$), indicating high linear relationship between the scoring systems. Cohen's Kappa coefficient of 0.76 demonstrates substantial agreement in classification of submissions into quality categories (excellent, good, fair, poor), surpassing the 0.60 threshold considered indicative of substantial agreement [20].

TABLE II
PERFORMANCE TESTING RESULTS

Metric	Value	Interpretation
Pearson Correlation	0.847	Strong positive correlation
Cohen's Kappa	0.76	Substantial agreement
Mean Absolute Error (MAE)	7.3 point	Average scoring difference
Root Mean Square Error (RMSE)	9.1 point	Score prediction error
Agreement within ±10 points	88%	High practical agreement
Agreement within ±5 points	64%	Moderate strict agreement

The Mean Absolute Error of 7.3 points (on a 0-100 scale) indicates that AI scores typically deviate less than 7.3 points from human expert scores. The 88% agreement rate within ±10 points demonstrates that the AI system provides practically equivalent assessments to human graders in the vast majority of cases. The 64% agreement within ±5 points show reasonable precision for fine-grained scoring.

F. AI Evaluation Quality Assessment

The quality of AI-generated evaluations was assessed by programming course instructors to ensure alignment with academic standards. The evaluation criteria included logical explanation accuracy, alignment with teaching materials, information clarity, educational value of feedback, and potential for accelerating evaluation processes.

Instructor feedback indicated that the AI explanations demonstrated good logical reasoning, though some cases could benefit from additional clarification. The system adequately accommodates general teaching materials, while specialized or customized topics may require supplementary explanations. The feedback provided was deemed sufficiently clear and met educational components enabling students to continue learning independently.

Notably, instructors confirmed that the AI system significantly helps accelerate evaluation processes, with the recommendation that detailed rubrics be created to ensure more explainable assessment results and reduce potential subjective bias in evaluation.

G. Token Usage and Cost Analysis

Token usage analysis was conducted using a complex Object-Oriented Programming problem to simulate real-world usage scenarios. The test involved a library management system implementation requiring approximately 200 lines of Java code, representing typical final exam complexity.

As shown in Table III, even for complex programming problems with lengthy solutions, the token usage remains well within DeepSeek-R1's maximum limit of 64,000 tokens per request. The cost analysis indicates that automated evaluation is economically viable for educational institutions, with each evaluation costing approximately $0.003 or about Rp 51.29.

TABLE III
TOKEN USAGE AND COST

Metric	Value
Input Token	2,141
Output Token	900
Total Token	3,041
Input cost (USD)	$0.001178
Output cost (USD)	$0.001970
Total Cost (USD)	$0.003148

IV. *CONCLUSION*

The research findings demonstrate that our web-based automated code evaluation system successfully addresses the challenges identified in programming education. Through integration of DeepSeek-R1 LLM with LangChain and Next.js frameworks, we developed a platform that transforms how programming assignments are assessed and feedback is delivered.

Our implementation reveals several key achievements. The evaluation demonstrates the system's readiness for institutional deployment. The substantial AI-human agreement (Cohen's Kappa = 0.76) and strong correlation (r = 0.847) validate assessment accuracy comparable to expert human graders. The also system maintains operational stability with a 5.7-second average response time, which proves acceptable for educational contexts where immediate feedback enhances learning outcomes. The throughput capacity of 10.86 requests per second, combined with successful scalability testing up to 150 concurrent users, indicates the platform can support typical classroom environments without performance bottlenecks.

Student acceptance proved encouraging, with a SUS score of 77 reflecting positive user experiences. Participants particularly valued the comprehensive feedback quality and fair scoring approach that awards partial credit rather than binary pass/fail results. The load time performance score of 78 further validates the technical implementation choices made during development.

From an institutional perspective, the economic analysis shows promising sustainability. Each evaluation costs approximately $0.003, making the technology financially viable for widespread educational adoption. This cost-effectiveness, combined with the significant time savings reported by instructors, presents a compelling case for implementation across programming curricula.

Perhaps most importantly, academic validation from programming instructors confirms that AI-generated feedback maintains educational standards while reducing evaluation subjectivity. The system consistently provides constructive guidance that helps students understand their mistakes and improve their coding skills, which represents the core pedagogical goal.

However, our work also identifies areas requiring continued attention. Interface design improvements, enhanced code editor functionality, and faster processing times emerged as priorities from user feedback. Additionally, implementing comprehensive classroom management features would better serve institutional needs.

This research contributes practical evidence that AI-powered programming assessment can successfully bridge the gap between educational demands and technological capabilities. The developed system not only demonstrates technical feasibility but also validates the educational value of automated evaluation when properly implemented with appropriate oversight and quality controls.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to Universitas Ma Chung for providing financial support through the Learning Innovation Grant (Hibah Inovasi Pembelajaran) that made this research possible. We also extend our appreciation to the Faculty of Technology and Design, Universitas Ma Chung, for providing the necessary infrastructure and academic environment to conduct this study. Special thanks to the students and faculty members who participated in the system testing and evaluation phases, whose valuable feedback contributed significantly to the improvement of the developed system.

AUTHORS CONTRIBUTION

Windra Swastika: Conceptualization, methodology design, project supervision, system architecture review, validation of results, review and editing of manuscript, funding acquisition.
Vincent Dwi Hartanto: System design and implementation, AI integration development, database design, user interface development, system testing and evaluation, data collection and analysis.
Paulus Lucky Tirma Irawan: Project administration, technical guidance, system validation, review of AI evaluation quality, mentorship throughout development process, manuscript review and editing.

COPYRIGHT

REFERENCES

[1] D. E. Knuth, The Art of Computer Programming, Volume 1: Fundamental Algorithms, 3rd ed. Addison-Wesley Professional, 1997.
[2] Abass, Olalere A., Samuel A. Olajıde, and Babafemi O. Samuel. "Development of web-based examination system using open source programming model." *Turkish Online Journal of Distance Education* 18.2 (2017): 30-42.
[3] Azaiz, Imen, Natalie Kiesler, and Sven Strickroth. "Feedback-generation for programming exercises with gpt-4." *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*. 2024. 31-37.
[4] Paiva, José Carlos, José Paulo Leal, and Álvaro Figueira. "Automated assessment in computer science education: A state-of-the-art review." *ACM Transactions on Computing Education (TOCE)* 22.3 (2022): 1-40.
[5] Hao, Qiang, et al. "Towards understanding the effective design of automated formative feedback for programming assignments." *Computer Science Education* 32.1 (2022): 105-127.
[6] Dewi, S. K. Artificial Intelligence. Graha Ilmu, 2018.
[7] Chen, Lijia, Pingping Chen, and Zhijian Lin. "Artificial intelligence in education: A review." *IEEE access* 8 (2020): 75264-75278.
[8] O. Zawacki-Richter, V. I. Marín, M. Bond, and F. Gouverneur, "Systematic Review of Research on Artificial Intelligence Applications in Higher Education – Where Are the Educators?," International Journal of Educational Technology in Higher Education, vol. 16, no. 1, p. 39, 2019.
[9] Vogelsang, Andreas, and Jannik Fischbach. "Using large language models for natural language processing tasks in requirements engineering: A systematic guideline." *Handbook on Natural Language Processing for Requirements Engineering*. Cham: Springer Nature Switzerland, 2025. 435-456.
[10] DeepSeek-AI. "DeepSeek Coder: Let the Code Write Itself." DeepSeek AI Blog, 2024. [Online]. Available: https://deepseek.com/blog/deepseek-coder
[11] Coello, Carlos Eduardo Andino, Mohammed Nazeh Alimam, and Rand Kouatly. "Effectiveness of chatgpt in coding: A comparative analysis of popular large language models." *Digital* 4.1 (2024): 114-125.
[12] Shakya, Ronas, Farhad Vadiee, and Mohammad Khalil. "A showdown of ChatGPT vs DeepSeek in solving programming tasks." *2025 International Conference on New Trends in Computing Sciences (ICTCS)*. IEEE, 2025.
[13] Ugale, Durgesh, et al. "Student Performance Prediction Using Data Mining Techniques." *International Research Journal of Engineering and Technology (IRJET)* 7.05 (2020).
[14] M. Chase, "LangChain: A Framework for Developing Applications Powered by Language Models," GitHub Repository, 2022. [Online]. Available: https://github.com/langchain-ai/langchain
[15] A. E. Kelly, R. A. Lesh, and J. Y. Baek, Eds., Handbook of Design Research Methods in Education: Innovations in Science, Technology, Engineering, and Mathematics Learning and Teaching. Routledge, 2008.
[16] J. Brooke, "SUS: A 'Quick and Dirty' Usability Scale," in Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and A. L. McClelland, Eds. London: Taylor & Francis, 1996.
[17] A. Bangor, P. T. Kortum, and J. T. Miller, "An Empirical Evaluation of the System Usability Scale," International Journal of Human-Computer Interaction, vol. 24, no. 6, pp. 574-594, 2008.
[18] K. M. Ala-Mutka, "A Survey of Automated Assessment Approaches for Programming Assignments," Computer Science Education, vol. 15, no. 2, pp. 83-102, 2005.
[19] N. Prather et al., "What Do We Think We Think We Are Doing? Metacognition and Self-Regulation in Programming," in Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER), 2020.
[20] J. R. Landis and G. G. Koch, "The Measurement of Observer Agreement for Categorical Data," Biometrics, vol. 33, no. 1, pp. 159-174, 1977.