ISSN: 2623-1662 (online), 2460-092X (printed)
DOI: https://doi.org/10.19109/jusifo.v11i2.30467
Vol. 11, No. 2 (2025), p.85-96

JUSIFO JURNAL SISTEM INFORMASI

Research Article [OPEN ACCESS]

Development of SEKAPAI: An AI-Based Scaffolding Platform for Programming Education

Rizki Hikmawan*, Dedi Rohendi, Jaka Septiadi, Muhamad Akda Fathul Barri

ABSTRACT

The rapid adoption of generative artificial intelligence in programming education has raised concerns regarding student over-dependence and the erosion of computational thinking skills. This study presents the design and internal validation of SEKAPAI, an AI-based scaffolding platform developed to support computational thinking while promoting responsible use of generative AI. Using an Agile-oriented Research and Development approach, SEKAPAI integrates three adaptive scaffolding modules—Solution Assessment, Code Assessment, and Free Interaction—to deliver context-aware feedback without providing direct solutions. System requirements were derived through stakeholder analysis and translated into a modular, web-based architecture supported by GPT-based services. Internal validation was conducted using comprehensive black-box testing to evaluate functional correctness, feedback behavior, and alignment with computational thinking components. The results indicate that SEKAPAI operates reliably across core system features and consistently implements progressive scaffolding strategies that regulate AI assistance. This study demonstrates how pedagogical scaffolding principles can be operationalized within AI-assisted learning systems and provides a technically feasible reference model for responsible AI integration in programming education.

Keyword: AI-based scaffolding, computational thinking, programming education

Received: July 26, 2025; Revised: October 05, 2025; Accepted: December 20, 2025

Corresponding Author: Rizki Hikmawan, Department of Information Technology and Systems Education, Universitas Pendidikan Indonesia, Indonesia, hikmariz@upi.edu
Authors: Dedi Rohendi, Department of Technology and Vocational Education, Universitas Pendidikan Indonesia, Indonesia, dedir@upi.edu; Jaka Septiadi, Department of Information Technology and Systems Education, Universitas Pendidikan Indonesia, Indonesia, jakaseptiadi@upi.edu; Muhamad Akda Fathul Barri, Department of Information Technology and Systems Education, Universitas Pendidikan Indonesia, Indonesia, akdafathul@upi.edu

1. INTRODUCTION

The integration of Artificial Intelligence (AI) in education has accelerated rapidly with the emergence of generative language models such as ChatGPT. These technologies have reshaped educational practices by enabling automated feedback, personalized learning support, and more intuitive interactions, particularly in academic writing and programming instruction (Kasneci et al., 2023; Rahman & Watanobe, 2023). In parallel, Computational Thinking has gained renewed prominence as a core 21st-century skill, extending beyond computer science into broader interdisciplinary learning contexts (Nouri et al., 2020).

Despite these advances, the increasing reliance on AI in educational settings presents substantial pedagogical challenges. Empirical evidence suggests that students often depend excessively on AI-generated responses, sometimes placing greater trust in tools such as ChatGPT than in validated academic sources (Yilmaz & Karaoglan Yilmaz, 2023). This overreliance risks weakening critical thinking, reducing engagement in structured problem-solving processes, and undermining key dimensions of computational thinking (Farrokhnia et al., 2024; Mhlanga, 2023). Furthermore, many AI-driven educational applications emphasize automation efficiency rather than pedagogical intentionality, raising concerns about learner autonomy and long-term learning outcomes.

Prior research has explored various instructional approaches, including gamified learning environments, adaptive systems, and AI-supported scaffolding strategies. While these studies highlight the importance of balancing AI assistance with learner independence (Dela Calzada, 2024; Liao et al., 2024), most remain conceptual or fragmented in practice. Existing platforms rarely provide an integrated mechanism that systematically regulates AI support to promote gradual skill development, particularly within programming education contexts that demand sustained computational thinking.

More critically, although scaffolding theory has been well established in educational research (Chi et al., 2001; Van de Pol et al., 2010) and has increasingly been examined in AI-supported learning, there remains a notable absence of a cohesive platform that operationalizes multi-level AI-based scaffolding. Specifically, no prior study has implemented a unified system integrating Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI) modules within an Agile development framework. This gap highlights the need for a practical and adaptive AI-based learning system capable of supporting learners while intentionally fading assistance as competence increases.

To address this gap, this study develops SEKAPAI, a web-based scaffolding platform powered by GPT and designed using Agile methodology. SEKAPAI integrates adaptive scaffolding modules—Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI)—to deliver contextual feedback aligned with computational thinking principles while progressively reducing AI intervention. This research evaluates SEKAPAI from two perspectives: (1) the effectiveness of its design in implementing responsible AI-assisted scaffolding and (2) its functional performance in supporting computational thinking development. Through this dual focus, the study contributes a pedagogically grounded and operational solution for responsible AI integration in programming education.

2. MATERIALS AND METHODS

2.1 Materials

This study employed a set of technical and computational materials to support the development and internal validation of the SEKAPAI platform. As a system design–oriented study in the field of information systems, the materials consisted primarily of software tools, artificial intelligence services, development infrastructure, and testing artifacts rather than physical instruments or human-subject datasets.

The platform was developed using modern web technologies to ensure scalability, maintainability, and cross-platform accessibility. The frontend interface was implemented using Next.js, while backend services were developed using Node.js and Express to coordinate communication between the client layer, database services, and AI components. User interface and interaction designs were prototyped using Figma to support a user-centered and mobile-first design approach.

Artificial intelligence functionalities within SEKAPAI were powered by the OpenAI GPT API (GPT-3.5 and GPT-4), which has been widely adopted in educational contexts for adaptive feedback and programming support. The AI models were accessed through structured system prompts tailored to each scaffolding module—Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI). Prompt configurations were designed to regulate the level of instructional support provided and to prevent direct solution disclosure, in line with recommended practices for responsible AI use in education (Liao et al., 2024).

For data management, PostgreSQL was used as the primary database to store interaction logs, system responses, and progress-related records generated during internal testing. The system infrastructure was hosted on Amazon Web Services (AWS), while system monitoring and error tracking during development were supported using CloudWatch and Sentry. Version control and collaborative development were managed through GitHub. System validation relied on predefined testing materials, including black-box testing scenarios, equivalence partitioning rules, and boundary value definitions. This testing strategy is commonly applied in software-based educational systems to assess functional correctness and output reliability without inspecting internal code structures (Gunawan et al., 2023).

2.2 Methods

This study adopts a Research and Development approach to design, implement, and internally validate an AI-based scaffolding platform. The methodological focus of this research is on system development, technical implementation, and functional validation, rather than large-scale empirical measurement of learning outcomes. To support iterative refinement and pedagogical alignment, the development process was guided by Agile principles, emphasizing flexibility, incremental delivery, and responsiveness to user feedback (Gheorghe et al., 2020; Larasati et al., 2021).

2.3 Research Design and Development Approach

The research design integrates three main components: (1) pedagogically informed system design, (2) Agile-based iterative development, and (3) internal system validation. This integration enables theoretical principles related to computational thinking and scaffolding to be operationalized into a functional digital system while maintaining practical feasibility.

The Agile-inspired approach was selected to enable continuous adaptation during development without rigid adherence to formal frameworks such as Scrum (Zhen, 2024). Development iterations were conducted in weekly cycles, allowing incremental enhancement of system features and early identification of design inconsistencies. This approach aligns with prior studies highlighting the effectiveness of Agile methods in educational system development and AI-supported learning environments (Gheorghe et al., 2020; Laval et al., 2021).

2.4 Requirements Analysis and User-Centered Design

Requirements analysis was conducted through structured interviews and classroom observations involving programming instructors and students as primary stakeholders. This phase aimed to identify learning difficulties, interaction preferences, and pedagogical constraints associated with programming education and AI usage. The analysis focused on four requirement dimensions: educational, user experience, pedagogical, and technical requirements (Moon et al., 2020; Yusoff et al., 2020).

Educational requirements emphasized the need to support computational thinking development while mitigating over-reliance on AI tools. User experience requirements addressed interface clarity, interaction flow, and accessibility. Pedagogical constraints were defined to prevent direct solution disclosure and encourage independent reasoning, consistent with concerns about AI dependency in educational contexts (Tiwari, 2023). Technical requirements specified system responsiveness, performance stability, and integration with AI services. The resulting requirements were translated into functional specifications and interaction flows to guide system design (Weintrop et al., 2021).

2.5 Agile-Inspired Development Process

The development of the SEKAPAI platform adopted an Agile-inspired iterative process to ensure flexibility, continuous refinement, and alignment between pedagogical objectives and technical implementation. Instead of strictly following a formal Agile framework, this study applied core Agile principles—incremental development, responsiveness to feedback, and iterative evaluation—to accommodate the evolving requirements of an AI-assisted learning system. The development cycle consisted of six recurring stages: requirements definition, design, implementation, testing, deployment, and review. During the design stage, user interface structures, system architecture, and AI interaction scenarios were specified for each scaffolding module, with explicit constraints imposed to prevent unrestricted AI assistance that could undermine learning objectives.

Implementation was carried out using a modular development strategy that enabled independent development and refinement of the Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI) modules. Each iteration incorporated internal testing feedback to verify functional correctness, interaction consistency, and adherence to predefined educational requirements. The review phase of each cycle emphasized both technical stability and pedagogical behavior, particularly the appropriateness of feedback depth and the gradual reduction of AI assistance. This iterative refinement process facilitated early identification of design inconsistencies, supported adaptive feature evolution, and reduced the risk of scope drift during extended development cycles, thereby providing a practical and controlled approach to operationalizing AI-based scaffolding within programming education.

2.6 System Architecture and Technical Implementation

SEKAPAI employs a modular, three-layer architecture consisting of a client layer, backend services, and cloud infrastructure. The client interface was developed using Next.js with Zustand for state management, enabling responsive interaction across devices. Backend services, implemented using Node.js and Express, function as an API gateway coordinating communication between the client interface, database services, and AI components.

AI-based feedback is generated through the OpenAI GPT API using structured prompts designed to support scaffolding principles while preventing solution dependency. Core backend components include a prompt engine, context manager, execution engine, and recommendation module, which collectively regulate interaction flow and adjust scaffolding intensity based on user behavior. This architecture operationalizes adaptive feedback aligned with computational thinking dimensions and the principles of scaffolding.

3. RESULTS AND DISCUSSION

3.1 Requirements Analysis

The requirements analysis was conducted through interviews with educational stakeholders and observations of students' programming learning patterns. This process yielded three interrelated categories of requirements—functional, non-functional, and educational—intended to ensure that SEKAPAI supports AI-assisted scaffolding while maintaining learning continuity and responsible AI use. Functionally, the platform must accept pseudocode and program code, generate AI-based feedback, support interactive dialogue, and manage user interaction history, which is essential for tracking learning progression and refining adaptive feedback strategies over time (Faber et al., 2024). Non-functional requirements emphasize responsiveness, usability, cross-platform accessibility, and stable performance under concurrent classroom use. Educational requirements focus on scaffolding-oriented learning support through constructive feedback and the gradual promotion of learner independence, explicitly addressing the risk of over-dependency associated with generative AI use in educational contexts (Rowlett & Corner, 2022).

To operationalize these functional, non-functional, and educational requirements, a scenario-based interaction model was defined to illustrate how the identified requirements are enacted within the system. Table 1 presents a representative interaction scenario in which a single programming task is addressed through the three scaffolding modules: Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI). The scenario demonstrates how functional requirements are implemented across modules while educational requirements are reflected through the regulation of AI assistance levels—ranging from conceptual reasoning without solution disclosure in SA, to implementation-focused feedback in CA, and controlled exploratory dialogue in FI. This staged and role-specific interaction clarifies how scaffolding intensity is progressively adjusted to foster learner independence while mitigating the risk of AI over-dependency, thereby validating the alignment between the identified requirements and the modular design of the SEKAPAI platform.

3.2 System Design Results

The system design phase resulted in a set of structured design artifacts that formalize the architecture, interaction model, and learning workflow of the SEKAPAI platform. The primary design outputs comprise the system architecture of SEKAPAI (Figure 1), the use case diagram of the SEKAPAI system (Figure 2), the activity diagram of the SEKAPAI learning workflow (Figure 3), and the sequence diagram of SEKAPAI modules (Figure 4). Collectively, these artifacts delineate the functional scope, interaction logic, and pedagogical flow of the platform across its three core modules: Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI).

Table 1. Scenario-based interaction requirements of the SEKAPAI platform

Role	**Scaffolding Module**	**Interaction Description**
System	---	Task as follows: Given a string containing lowercase letters and parentheses, remove all parentheses and return the modified string. Example: input (a(b)c)d becomes abcd.
User	SA	Please evaluate the following solution: Use a count variable to track the number of opening parentheses. Iterate through each character. If character is '(', increment count. If ')', decrement count. At the end of iteration, if count = 0, then string is valid. If not, return false.
Assistant	SA	Okay, I've reviewed your proposed solution. It seems like you're on the right track with using a counter to keep track of the balance between opening and closing parentheses. However, let's think a bit more deeply about potential edge cases and scenarios. • What happens if you encounter a closing parenthesis before an opening one? Does your current approach handle this situation correctly? • Could you provide an example string that would cause your solution to return an incorrect result? Thinking about such examples can help refine your algorithm. • How would you handle other characters in the string besides parentheses? Should they be ignored, or could they potentially impact the validity of the string in some way? Consider these questions, and let me know if you'd like to discuss further!
User	CA	Please evaluate the following code: public class RemoveParentheses {public static String remove(String s) {Stack<Character> stack = new Stack<>();StringBuilder sb = new StringBuilder();for (char c : s.toCharArray()) {if (c == '(') stack.push(c);else if (c == ')') stack.pop();else sb.append(c);}return sb.toString();}
Assistant	CA	This code uses a stack to remove parentheses, but does not yet handle cases of unbalanced parentheses.
User	FI	What if there are unbalanced parentheses in the input?
Assistant	FI	Solution for handling mismatch: if (!stack.isEmpty() && stack.peek() == '(') { stack.pop();} else { throw new IllegalArgumentException("Invalid input");}

At the architectural level, the system architecture illustrates the overall structural organization of SEKAPAI, including the client layer, backend services, and AI integration components. This structure defines how user interactions are managed through the frontend interface, processed by backend services, and supported by GPT-based AI feedback mechanisms. The layered design promotes modularity, scalability, and maintainability while enabling the implementation of adaptive scaffolding strategies within the learning environment.

From an interaction perspective, the Use Case Diagram specifies the relationships between system actors and platform functionalities. Two primary actor groups are identified: students and administrators. Students are provided access to all learning modules (SA, CA, and FI), profile management features, and interaction history, whereas administrators are responsible for system configuration and maintenance. This interaction model supports flexible learning pathways and systematic progress monitoring, which are essential for personalized learning support in digital education environments (Raghavan et al., 2021).

The Activity Diagram of the SEKAPAI learning workflow represents the end-to-end learning process experienced by users. The workflow begins with user authentication and dashboard access, followed by module selection, input submission, AI-based analysis, feedback generation, interaction logging, progress tracking, and recommendation delivery. This activity flow operationalizes adaptive learning paths and real-time feedback mechanisms, thereby supporting individualized learning experiences and the development of computational thinking skills (Sayed et al., 2022).

Building on this workflow representation, the Sequence Diagram details the temporal interaction between users, system components, and GPT-based AI services. It illustrates how user inputs are transmitted through backend services, processed by the GPT API, and returned as scaffolded feedback across the SA, CA, and FI modules. This interaction sequence ensures consistency in inquiry handling, code evaluation, and exploratory dialogue while preserving contextual continuity throughout the learning process.

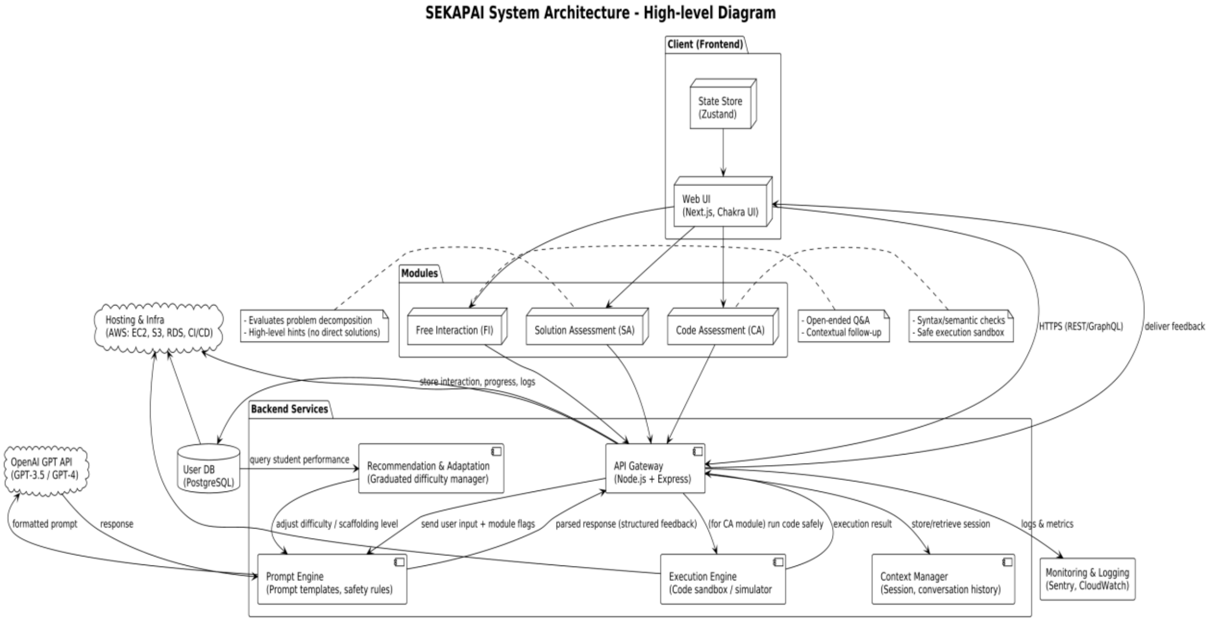

Figure 1. System architecture of SEKAPAI

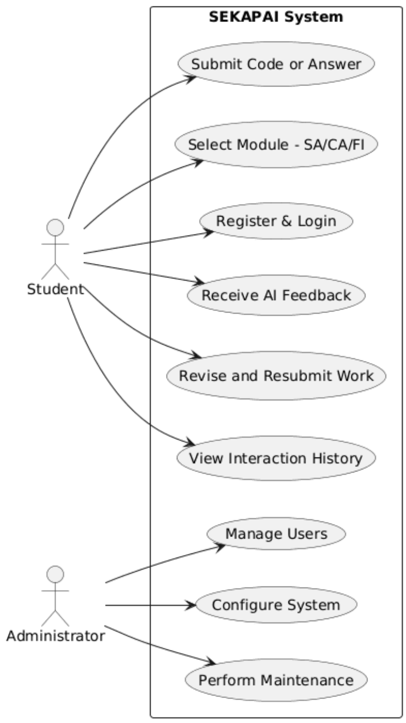

Figure 2. Use case diagram of the SEKAPAI system

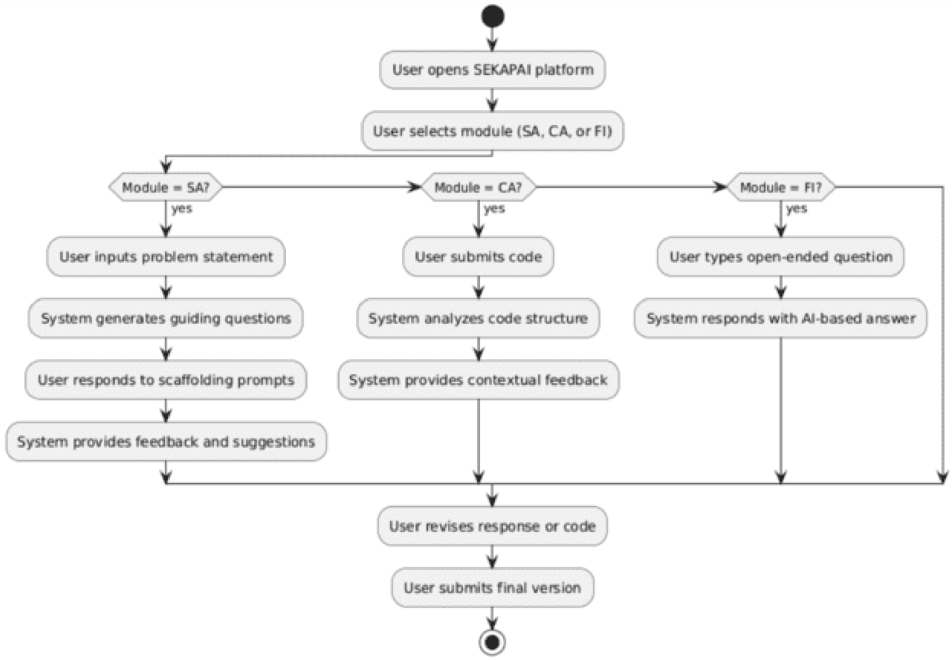

Figure 3. Activity diagram of SEKAPAI learning workflow

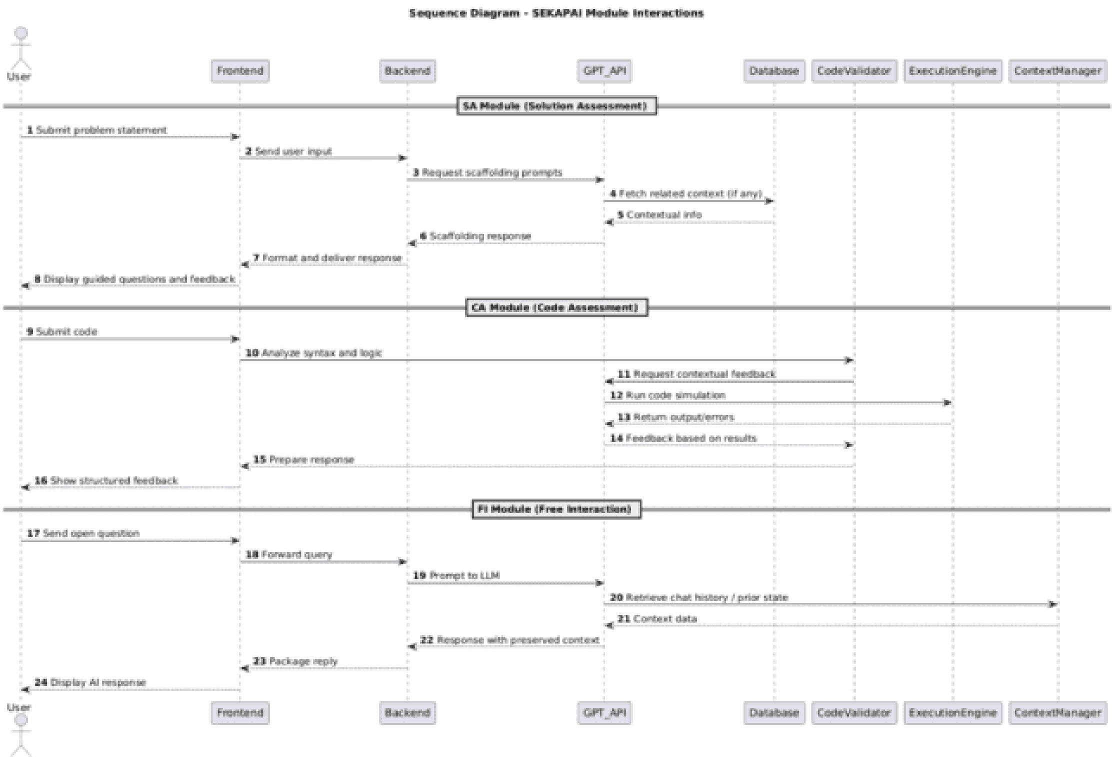

Figure 4. Sequence diagram of SEKAPAI modules

3.3 Validation Results

System validation results were obtained through comprehensive black-box testing, in which each scenario evaluated system functionality, feedback quality, and alignment with computational thinking development. Testing was organized around core system features and scaffolding modules, with expected and actual outcomes documented and summarized in Table 2. The results indicate that essential system functions (e.g., sign up and sign in) and module-level behaviors across Solution Assessment (SA), Code

Assessment (CA), and Free Interaction (FI) consistently achieved a Pass status. Importantly, the educational impact of each scenario is explicitly mapped to computational thinking components—such as decomposition, abstraction, debugging, and pattern recognition—as well as to responsible AI-use behaviors, including the redirection of direct solution-seeking attempts toward scaffolded guidance.

Beyond functional correctness, the validation results demonstrate the system's adaptive scaffolding capabilities. The testing outcomes show that SEKAPAI is able to differentiate between novice and intermediate input cases, regulate feedback depth accordingly, and promote learner independence through progressive adjustment of assistance. In particular, the Code Assessment module achieved over 95% functional accuracy in syntax recognition and response delivery, indicating technical robustness. Collectively, these findings provide evidence of the platform's technical feasibility and pedagogical alignment, while establishing a foundation for subsequent pilot testing with 5–10 students using standardized usability and user experience instruments such as SUS and UEQ.

Table 2. Black-box testing results of the SEKAPAI platform

ID	Feature / Module	Test Description (Scenario)	Expected Result	Actual Result	Status	Educational Impact
TC-SGNUP	Sign Up	User registration using a valid email address	User account is created successfully	Account created and confirmation email delivered	Pass	Enables personalized interaction tracking
TC-SGNIN	Sign In	User login with valid credentials	User successfully accesses the dashboard	Dashboard loaded and session activated	Pass	Supports continuity of learning activities
TC-SA-01	Solution Assessment (SA)	Submission of pseudocode for a sorting algorithm	System provides conceptual feedback on solution logic	Constructive feedback on algorithmic structure generated	Pass	Supports decomposition and abstraction
TC-SA-02	Solution Assessment (SA)	Submission of an incomplete algorithm description	System prompts clarification while offering guidance	Scaffolded feedback provided without disclosing solutions	Pass	Encourages reflective and independent thinking
TC-CA-01	Code Assessment (CA)	Submission of code containing syntax errors	System detects errors and provides improvement suggestions	Relevant diagnostic feedback generated	Pass	Enhances debugging skills
TC-CA-02	Code Assessment (CA)	Submission of functionally correct but inefficient code	System provides efficiency-oriented recommendations	Constructive feedback on code optimization delivered	Pass	Develops algorithmic efficiency awareness
TC-FI-01	Free Interaction (FI)	Conceptual Q&A related to programming topics	System generates contextual and educational responses	Relevant explanations supporting conceptual understanding	Pass	Supports pattern recognition
TC-FI-02	Free Interaction (FI)	Attempt to obtain direct assignment solutions	System redirects interaction toward scaffolded learning	Direct solutions refused; guided learning prompts provided	Pass	Mitigates AI dependency and academic misconduct

3.4 Deployment and Internal Review Results

Following the completion of black-box testing, the SEKAPAI platform was deployed to a production environment using Amazon Web Services (AWS). The deployment process involved environment configuration, PostgreSQL database migration, OpenAI API integration with rate limiting, custom domain and SSL certificate setup, as well as the implementation of monitoring and logging mechanisms. Deployment results indicate that the system achieved stable operational performance, with recorded uptime reaching 99.9% and average GPT API response times remaining below 500 milliseconds. These outcomes demonstrate the platform's readiness to deliver reliable AI-assisted educational services under real-world operational conditions.

In parallel with deployment, an internal review was conducted through structured trials by the development team to assess system behavior across technical, pedagogical, and user experience dimensions. The review observations indicate that SEKAPAI performed consistently across its core modules, delivering timely and contextually appropriate AI-based feedback while maintaining the intended scaffolding constraints. The interaction flow across modules was found to support guided problem-solving without disclosing direct solutions, thereby preserving pedagogical integrity. Overall, the deployment and internal review results confirm that SEKAPAI is technically stable, operationally viable, and aligned with its design objective of supporting responsible AI-assisted learning prior to empirical evaluation with student users.

3.5 Discussion

The development and internal validation of the SEKAPAI platform represent a strategic response to the increasing integration of artificial intelligence in programming education, particularly in addressing the pedagogical risks associated with students' over-dependence on generative tools such as ChatGPT. This study demonstrates that, when supported by intentional scaffolding mechanisms—namely the Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI) modules—AI can be repositioned from a substitute for student reasoning to a dynamic form of cognitive support. The system-generated feedback was found to be adaptive and contextually relevant, aligning with core components of computational thinking, including decomposition, abstraction, pattern recognition, and algorithmic design.

The findings further indicate that an iterative Agile development process plays a critical role in maintaining both pedagogical alignment and system usability in AI-based learning environments. Unlike linear development models, the Agile approach enabled continuous refinement based on user interaction and internal testing, contributing to the reliability of system features and the relevance of AI-generated feedback. This observation is consistent with prior studies reporting that Agile methodologies in educational system development enhance learner engagement and motivation (Laval et al., 2021; Saputra et al., 2025).

SEKAPAI's design choice to implement graduated feedback while deliberately avoiding direct solution provision reflects current perspectives in AI ethics and pedagogy. In line with the framework proposed by Liao et al. (2024), which emphasizes AI systems that support rather than replace learner thinking, SEKAPAI encourages reflection, revision, and problem-solving through guided prompts. This approach is central to fostering learner independence. Similarly, the findings align with Faber et al. (2024), who reported that adaptive scaffolding mechanisms improve learner engagement and reduce cognitive overload—effects that were also observed during the internal validation phase of SEKAPAI.

The alignment between SEKAPAI's system architecture and educational theory further supports its long-term viability. The integration of prompt engineering, modular feedback loops, and interaction history tracking enables a responsible AI design that promotes learner autonomy over time. Unlike conventional black-box AI systems, where interaction is often passive and pedagogical transparency is limited, SEKAPAI embeds instructional logic directly into its interaction design, ensuring that AI assistance remains transparent, adaptive, and pedagogically grounded. Overall, the findings suggest that generative AI can be pedagogically meaningful when implemented through intentional scaffolding and iterative refinement. Beyond its immediate functional value, SEKAPAI offers a transferable design framework for learning contexts that require critical thinking and iterative problem-solving; however, further empirical evaluation in authentic educational settings is required to assess scalability, long-term effectiveness, and cross-disciplinary applicability.

4. CONCLUSION

This study presented the design, development, and internal validation of SEKAPAI, an AI-based scaffolding platform intended to support computational thinking development in programming education while mitigating excessive reliance on generative AI tools. Developed using an Agile-oriented Research and Development approach, SEKAPAI integrates three adaptive scaffolding modules—Solution Assessment (SA), Code Assessment (CA), and Free Interaction (FI)—to deliver context-aware feedback aligned with established computational thinking components. The internal validation results confirm that the system operates reliably and successfully implements progressive scaffolding through controlled AI assistance rather than direct solution provision.

From a systems perspective, SEKAPAI demonstrates how pedagogical scaffolding principles can be explicitly embedded into the architecture and interaction design of an AI-assisted learning platform. The modular, web-based architecture supports extensibility and potential integration with existing learning management systems, while the adaptive fading mechanism offers a concrete strategy for balancing AI support and learner autonomy. These characteristics position SEKAPAI as a technically feasible and pedagogically grounded reference model for responsible AI integration in programming-related learning contexts.

Nevertheless, this research is intentionally limited to system design and internal validation. No empirical evaluation involving real student users has been conducted at this stage. Consequently, learning effectiveness, usability, and user experience outcomes cannot yet be inferred. Future research will therefore focus on pilot implementation in authentic educational settings, where usability and user experience will be systematically evaluated using standardized instruments such as the System Usability Scale (SUS) and User Experience Questionnaire (UEQ). Further extensions may include learning analytics, instructor-facing dashboards, and adaptive feedback calibration mechanisms to support scalability and cross-disciplinary application.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

Chi, M. T. H., Siler, S. A., Jeong, H., Yamauchi, T., & Hausmann, R. G. (2001). Learning from human tutoring. *Cognitive Science*, *25*(4), 471–533. https://doi.org/10.1207/S15516709COG2504_1

Dela Calzada, K. P. (2024). Anti-dependency teaching strategy for innovation in the age of ai among technology-based students. *Environment and Social Psychology*, *9*(8). https://doi.org/10.59429/ESP.V9I8.3026

Faber, T. J. E., Dankbaar, M. E. W., van den Broek, W. W., Bruinink, L. J., Hogeveen, M., & van Merriënboer, J. J. G. (2024). Effects of adaptive scaffolding on performance, cognitive load and engagement in game-based learning: a randomized controlled trial. *BMC Medical Education*, *24*(1), 943-. https://doi.org/10.1186/S12909-024-05698-3

Farrokhnia, M., Banihashem, S. K., Noroozi, O., & Wals, A. (2024). A swot analysis of chatgpt: implications for educational practice and research. *Innovations in Education and Teaching International*, *61*(3), 460–474. https://doi.org/10.1080/14703297.2023.2195846

Gheorghe, A.-M., Gheorghe, I. D., & Iatan, I. L. (2020). Agile software development. *Informatica Economica*, *24*(2/2020), 90–100. https://doi.org/10.24818/issn14531305/24.2.2020.08

Gunawan, R., Wibisono, Y. P., Primasari, C. H., Budiyanto, D., & Cininta, M. (2023). Blackbox testing on virtual reality gamelan saron using equivalence partition method. *Jurnal Buana Informatika*, *14*(01), 11–19. https://doi.org/10.24002/JBI.V14I01.6606

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., … Kasneci, G. (2023). Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and Individual Differences*, *103*, 102274. https://doi.org/10.1016/J.LINDIF.2023.102274

Larasati, I., Yusril, A. N., & Zukri, P. Al. (2021). Systematic literature review analisis metode agile dalam pengembangan aplikasi mobile. *Sistemasi: Jurnal Sistem Informasi*, *10*(2), 369–380. https://doi.org/10.32520/STMSI.V10I2.1237

Laval, J., Fleury, A., Karami, A. B., Lebis, A., Lozenguez, G., Pinot, R., & Vermeulen, M. (2021). Toward an innovative educational method to train students to agile approaches in higher education: the a.l.p.e.s. *Education Sciences*, *11*(6), 267. https://doi.org/10.3390/EDUCSCI11060267

Liao, J., Zhong, L., Zhe, L., Xu, H., Liu, M., & Xie, T. (2024). Scaffolding computational thinking with chatgpt. *IEEE Transactions on Learning Technologies*, *17*, 1668–1682. https://doi.org/10.1109/TLT.2024.3392896

Mhlanga, D. (2023). Open ai in education, the responsible and ethical use of chatgpt towards lifelong learning. *SSRN Electronic Journal*. https://doi.org/10.2139/SSRN.4354422

Moon, J., Do, J., Lee, D., & Choi, G. W. (2020). A conceptual framework for teaching computational thinking in personalized oers. *Smart Learning Environments 2020 7:1*, *7*(1), 6-. https://doi.org/10.1186/S40561-019-0108-Z

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in k-9. *Education Inquiry*, *11*(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844

Raghavan, S., M, S., S, S., & Devi M, Dr. D. (2021). Student support system an one stop portal. *International Journal of Advanced Research in Science, Communication and Technology*, 66–71. https://doi.org/10.48175/IJARSCT-1211

Rahman, M. M., & Watanobe, Y. (2023). Chatgpt for education and research: opportunities, threats, and strategies. *Applied Sciences*, *13*(9), 5783. https://doi.org/10.3390/APP13095783

Rowlett, P., & Corner, A. S. (2022). Flexible, student-centred remote learning for programming skills development. *International Journal of Mathematical Education in Science and Technology*, *53*(3), 619–626. https://doi.org/10.1080/0020739X.2021.1989067

Saputra, J. P. B., Prabowo, H., Gaol, F. L., & Hertono, G. F. (2025). Development of gamification-based learning management system (lms) with agile approach and personalization of fslsm learning style to improve learning effectiveness. *Journal of Applied Data Sciences*, *6*(1), 714–725. https://doi.org/10.47738/JADS.V6I1.486

Sayed, W. S., Noeman, A. M., Abdellatif, A., Abdelrazek, M., Badawy, M. G., Hamed, A., & El-Tantawy, S. (2022). AI-based adaptive personalized content presentation and exercises navigation for an effective and engaging e-learning platform. *Multimedia Tools and Applications*, *82*(3), 3303–3333. https://doi.org/10.1007/S11042-022-13076-8

Tiwari, R. (2023). The integration of ai and machine learning in education and its potential to personalize and improve student learning experiences. *International Journal of Scientific Research in Engineering and Management*, *07*(02). https://doi.org/10.55041/IJSREM17645

Van de Pol, J., Volman, M., & Beishuizen, J. (2010). Scaffolding in teacher–student interaction: a decade of research. *Educational Psychology Review*, *22*(3), 271–296. https://doi.org/10.1007/S10648-010-9127-6

Weintrop, D., Wise Rutstein, D., Bienkowski, M., & McGee, S. (2021). Assessing computational thinking: an overview of the field. *Computer Science Education*, *31*(2), 113–116. https://doi.org/10.1080/08993408.2021.1918380

Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (ai)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, *4*, 100147. https://doi.org/10.1016/J.CAEAI.2023.100147

Yusoff, K. M., Ashaari, N. S., Wook, T. S. M. T., & Ali, N. M. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. *International Journal of Advanced Computer Science and Applications*, *11*(3), 244–253. https://doi.org/10.14569/IJACSA.2020.0110329

Zhen, Z. (2024). Research and practice of agile software development methods. *Applied and Computational Engineering*, *114*(1), 186–190. https://doi.org/10.54254/2755-2721/2024.18284