JEULAMÈ: Journal of Education, Language, Arts, Multicultural, and Expression)
Volume 1, Number 2, 2026 pp. 45-58
P-ISSN: 2355-004X E-ISSN: 2502-6801
Open Access: https://ejournal.bbg.ac.id/jeulame

SPRINTA-BSD: A CODING LEARNING MODEL BASED ON COMPUTATIONAL THINKING, PROJECTS, AND REFLECTION IN DIGITAL LEARNING

Amilia Khasanah Marzuki*[1] and Adi Bandono[2]

[1]*Universitas Adi Buana Surabaya, Indonesia*
[2] *Universitas Adi Buana Surabaya, Indonesia*

* Corresponding Author: *250020069@student.unipasby.ac.id*

ARTICLE INFO

Article history:
Received: May 16, 2026
Revised: Jun 20, 2026
Accepted: Jul 18, 2026
Available online: Aug 31, 2026

Keywords:
AI education, coding, computational thinking, digital portfolio, project-based learning, SPRINTA-BSD

ABSTRACT

Coding learning in schools should not be understood merely as syntax writing practice, but as a learning space to develop computational thinking, problem solving, collaboration, creativity, and reflection. This study aims to formulate and analyze the SPRINTA-BSD coding learning model based on the app.bsd.application platform. The study employed a qualitative approach through a structured conceptual review. The primary sources were the SPRINTA-BSD model development manuscript and an examination of platform features, while supporting sources included national and international literature on coding education, computational thinking, project-based learning, AI education, digital assessment, and flexible learning. The analysis was conducted through concept reduction, thematic coding, syntax mapping, and pedagogical synthesis. The findings indicate that SPRINTA-BSD consists of six main stages: Sign In and Stimulation, Project Orientation and Problem Mapping, Resource Reading and Guided Exploration, Interactive Coding and Idea Building, Networked Technology Action, and Assessment, Reflection, and Advancement. The novelty of the model lies in its integration of learning syntax, platform features, teacher activities, student activities, process assessment, digital portfolios, sandbox exploration, feedback, and reflection into a coherent learning flow. The model strengthens coding learning by positioning errors as learning resources rather than failures. SPRINTA-BSD is also aligned with the Naval Collaboration Flexible Learning (NCFL) concept in terms of collaboration, flexibility, authentic learning experiences, and planned instructional design. This study concludes that SPRINTA-BSD has the potential to become a contemporary coding learning model; however, its effectiveness needs to be further examined through expert validation and classroom implementation.

This is an open access article under the CC BY-NC license.
Copyright © 2026 by Author. Published by Universitas Bina Bangsa Getsempena

INTRODUCTION

The development of artificial intelligence, automation, data, and digital platforms is changing the competencies that students need. It is no longer enough for students to be mere users of technology; they need to understand how technology works, recognize problems, devise solution steps, test solutions, and evaluate the impact of technology on life. In this context, coding education holds a strategic position because it can serve as a means of training logical, systematic, creative, and reflective thinking.

The idea of computational thinking has long been understood as the ability to solve problems, design systems, and understand human behavior by drawing on fundamental concepts of computer science (Wing, 2006). In education, computational thinking is not only related to the ability to write programs, but also encompasses problem decomposition, pattern recognition, abstraction, algorithm design, testing, debugging, and reflection (Brennan & Resnick, 2012; Shute et al., 2017). Therefore, good coding instruction should not stop at whether a program runs successfully, but should also assess how students understand problems, develop strategies, correct errors, and explain their thinking processes.

Various studies show that coding and computational thinking are increasingly relevant to cross-curricular education. The review by Mills et al. (2025) indicates that learning coding and computational thinking can support problem solving, creativity, critical thinking, student agency, and social skills. Rao and Bhagat (2024) also emphasize that strengthening computational thinking requires alignment among digital tools, pedagogical strategies, and assessment practices. This suggests that a learning platform should not be treated merely as a place to upload materials, but should be designed as a learning ecosystem that connects goals, activities, feedback, and assessment.

On the other hand, coding instruction in schools often faces several obstacles. First, instruction still tends to be oriented toward imitating syntax, so students do not always understand the reasoning behind the code they write. Second, digital learning sometimes merely transfers conventional tasks onto a platform without designing an adequate pedagogical flow. Third, mistakes or errors are still often regarded as failures, even though in coding an error is an opportunity to reread the problem, test hypotheses, discuss, and improve the solution. Fourth, assessment often focuses more on the final product than on the thinking process and the improvements students make.

The SPRINTA-BSD learning model was developed to address these problems. The model uses the app.bsd.application platform as a learning environment that integrates a classroom, project review, objective panel, resources, coding editor, output panel, sandbox,

portfolio, classroom chat, and progress tracking. SPRINTA-BSD comprises six learning stages, namely Sign In and Stimulation, Project Orientation and Problem Mapping, Resource Reading and Guided Exploration, Interactive Coding and Idea Building, Networked Technology Action, and Assessment, Reflection, and Advancement.

The novelty of this study lies in formulating a platform-based coding learning model that connects pedagogical syntax, digital features, teacher activities, student activities, process assessment, portfolios, sandbox, feedback, and reflection within a single learning flow. The model not only draws inspiration from project-based learning and computational thinking, but also links them to the idea of Naval Collaboration Flexible Learning (NCFL), which emphasizes collaboration, flexibility, authentic learning experiences, and a planned learning system (Bandono, 2020a, 2020b).

Based on this background, this study aims to: (1) formulate the conceptual structure of the SPRINTA-BSD model; (2) analyze the relationship between each stage of the model and the principles of computational thinking, project-based learning, and flexible learning; and (3) identify the implications, challenges, and recommendations for implementing the model in school coding instruction.

RESEARCH METHODOLOGY

This study uses a qualitative approach through a structured conceptual review. A conceptual review was chosen because the aim of the study is not to test the effectiveness of the model quantitatively, but to formulate, organize, and analyze the pedagogical foundations of the SPRINTA-BSD model. Accordingly, this article does not present claims of improved learning outcomes based on experimental data, but presents a scientific argument concerning the rationale of the model, its novelty position, and the implications of its implementation.

The primary sources of the study are the SPRINTA-BSD coding learning model development manuscript and an examination of the features of app.bsd.application that are relevant to the learning process, such as the classroom, project review, objective panel, resources, coding editor, output panel, sandbox, portfolio, classroom chat, and progress tracking. Supporting sources consist of journal articles, proceedings, academic books, and scientific documents that discuss coding education, computational thinking, project-based learning, AI education, digital assessment, collaborative learning, and flexible learning.

The supporting literature was selected using the following criteria: (1) it addresses coding education, computational thinking, AI education, project-based learning, or digital learning; (2) it is relevant to the context of primary, secondary, or instructional design

education; (3) it was published mainly within the last five years for empirical articles and recent reviews, while retaining several classic references that serve as conceptual foundations; and (4) it has traceable bibliographic information. Literature that is not directly related to education, discusses only technical aspects of programming without learning implications, or does not provide a clear publication identity was not used as a primary source of analysis.

The analysis was conducted in four stages. First, concept reduction, namely identifying the main concepts from the SPRINTA-BSD manuscript and the supporting literature. Second, thematic coding, namely grouping the concepts into the themes of computational thinking, project-based learning, AI literacy, process assessment, digital portfolios, reflection, and flexible learning. Third, syntax mapping, namely linking the SPRINTA-BSD stages with platform features, teacher activities, student activities, skill indicators, and learning evidence. Fourth, pedagogical synthesis, namely formulating the contributions, limitations, and implementation recommendations of the model.

The validity of the review was maintained through triangulation of conceptual sources, consistency of argumentation, and traceability of citations. Triangulation was carried out by comparing the principles of SPRINTA-BSD with international literature on computational thinking and AI education, national literature on coding education and computational thinking, and the NCFL idea as a collaborative-flexible learning framework. A limitation of this study is the absence of expert validation and empirical classroom trials, so the findings of this article should be viewed as a conceptual basis for subsequent development research.

RESULT AND DISCUSSION

The Need for a Structured Coding Learning Model

The results of the review show that coding instruction requires a more comprehensive design than mere syntax demonstrations and task assignments. Weintrop et al. (2016) emphasize that computational thinking in a learning context needs to include data practices, modeling and simulation, computational problem solving, and systems thinking. In other words, coding should be positioned as an activity of thinking and making, not merely a technical activity in front of a screen.

The review by Rao and Bhagat (2024) reinforces the view that the effectiveness of computational thinking is influenced by the choice of tools, learning strategies, and assessment. This finding is important because much digital learning still emphasizes the use of platforms without designing a clear relationship between platform features and

pedagogical goals. SPRINTA-BSD attempts to bridge this gap by giving every platform feature an explicit learning function.

The need for a structured model is also reinforced by the Indonesian context. Fitriyah et al. (2024) show that Indonesian students' computational thinking abilities still need to be strengthened through curriculum support, educator training, and inclusive learning environments. Nur and Nurhafidzah (2025) also found that coding instruction can affect computational thinking, creativity, motivation, problem solving, and digital literacy, but its implementation faces obstacles in infrastructure, understanding of abstract concepts, and teacher readiness. These conditions indicate that a coding learning model needs to specify the learning flow, mitigation strategies, and process assessment more clearly.

The Novelty Position of SPRINTA-BSD

SPRINTA-BSD is not merely the name of a sequence of learning stages, but a pedagogical design that positions the digital platform as a learning ecosystem. Its novelty can be seen in the integration of problem orientation, resource exploration, coding practice, work modification, portfolios, feedback, and reflection. This novelty position is presented in Table 1.

Table 1. Novelty Position of SPRINTA-BSD Compared with Previous Approaches

Approach	Main Focus	Strengths	Novelty in SPRINTA-BSD
Computational thinking	Decomposition, patterns, abstraction, algorithms, debugging, reflection	Strengthens thinking processes and problem solving (Shute et al., 2017; Weintrop et al., 2016; Wing, 2006)	CT is translated into a sequence of platform activities: objective, coding, output, debugging, portfolio, and reflection.
Project-based learning	Learning through projects, products, collaboration, and problem solving	Encourages active learning and complex problem solving (Shin et al., 2021)	Projects are not only a final assignment but become a flow from problem orientation to sandbox development.
AI education	AI literacy, understanding of AI concepts, AI learning tools, and technology ethics	Supports AI literacy, motivation, ethical reflection, and problem solving (Celik, 2023; Chiu et al., 2023; Lee & Kwon, 2024; Long & Magerko, 2020; Touretzky et al., 2019; Yim & Su, 2025)	Coding material is linked with AI, bias, digital ethics, creativity, and the impact of technology.

Approach	Main Focus	Strengths	Novelty in SPRINTA-BSD
NCFL	Collaboration, flexibility, authentic experience, planned learning system	Emphasizes adaptive and collaborative learning design (Bandono, 2020a, 2020b; Bandono et al., 2020)	Collaborative-flexible principles are applied to coding instruction based on the classroom, chat, progress, and portfolio.
SPRINTA-BSD	Coding model based on platform, projects, portfolio, feedback, and reflection	Combines thinking processes, digital practice, collaboration, and process assessment	Makes errors, feedback, and revision the main part of the coding learning cycle.

Operational Syntax of SPRINTA-BSD

SPRINTA-BSD consists of six sequential stages. Each stage has a pedagogical objective, platform features, teacher activities, student activities, computational thinking indicators, and learning evidence. This mapping is important so that teachers do not merely use the platform technically, but also understand the pedagogical function of each activity.

Table 2. Operational Syntax of the SPRINTA-BSD Model

Stage	Pedagogical Objective	Platform Features	Teacher Activities	Student Activities	Learning Evidence
S: Sign In and Stimulation	Builds technical readiness, curiosity, and initial problem orientation.	Login, classroom, opening material.	Ensures access, sparks the problem context, and communicates the objectives.	Logs in, enters the classroom, reads the trigger, and connects the problem with experience.	Access readiness, initial responses, and question notes.
P: Project Orientation and Problem Mapping	Trains students to understand the problem before writing code.	Project review, objective panel, target output.	Guides students to read the objective, input, process, output, and success criteria.	Maps the project objective, initial logic, and solution steps.	Problem map, objective checklist, and logic design.
R: Resource Reading and Guided Exploration	Strengthens source literacy, coding concepts, and accuracy in following instructions.	Resources, PDF/material , code examples, syntax tags.	Gives directions for reading sources, marking important concepts, and connecting materials with the project.	Reads materials, notes syntax, identifies concepts, and asks questions.	Concept notes, source summaries, and a list of key syntax.
I: Interactive Coding and Idea Building	Develops algorithms, debugging, creativity, and collaboration.	Coding editor, output panel, classroom chat.	Facilitates practice, poses guiding questions, and directs debugging.	Writes code, runs the program, reads errors, discusses, and improves solutions.	Running code, debugging log, output, and discussion.

Stage	Pedagogical Objective	Platform Features	Teacher Activities	Student Activities	Learning Evidence
NTA: Networked Technology Action	Encourages independence, modification, innovation, and ownership of work.	Sandbox, portfolio, share/review project.	Provides further challenges and facilitates the development of work.	Modifies the project, adds features, saves work, and prepares a presentation.	Modified work, portfolio, and development notes.
A: Assessment, Reflection, and Advancement	Makes assessment a process of continuous improvement.	Progress tracking, feedback, classroom chat, portfolio.	Assesses process and product, gives specific feedback, and determines follow-up.	Revises work, writes reflections, receives feedback, and designs improvements.	Rubric, teacher comments, reflection, revision, and follow-up plan.

Pedagogical Meaning of Each Stage

The Sign In and Stimulation stage functions to build learning readiness. Technical readiness such as logging in and entering the classroom is important because access barriers often disrupt the flow of digital learning. However, this stage must not stop at technical activities. Teachers need to connect the coding topic with real problems so that students understand why the project needs to be done.

The Project Orientation and Problem Mapping stage is an important distinguishing feature between imitation-based coding instruction and understanding-based coding instruction. At this stage, students are guided to read the project review, understand the target output, and map the input, process, and output. These activities develop decomposition, abstraction, and algorithm design, which are core parts of computational thinking (Shute et al., 2017; Weintrop et al., 2016).

The Resource Reading and Guided Exploration stage positions digital literacy as a prerequisite for coding practice. Many students fail to complete projects not because they cannot type code, but because they do not read the instructions, do not understand the function of the syntax, or do not connect the learning resources with the project problem. Therefore, reading resources and exploring instructions need to be an explicit part of the learning syntax.

The Interactive Coding and Idea Building stage is the center of coding activity. At this stage, students write code, run programs, view output, find errors, discuss, and improve their solutions. This is where errors are positioned as a learning resource. This perspective is aligned with project-based learning because students learn through building artifacts, testing ideas, and improving results (Shin et al., 2021).

The Networked Technology Action stage extends activity from guided projects to independent modification. Through the sandbox and portfolio, students not only complete tasks but also develop digital works that express identity, creativity, and agency. This idea

is close to computational action, namely computing education that not only teaches concepts but also gives students space to create works that have an impact on their lives and communities (Tissenbaum et al., 2019).

The Assessment, Reflection, and Advancement stage closes the learning cycle through assessment, feedback, revision, and reflection. Assessment is directed not only at the final product but also at the process of reading instructions, constructing logic, debugging, collaborating, receiving feedback, and improving work. In this way, coding instruction becomes a more humane process because it recognizes mistakes as part of intellectual growth.

3.5. Integration of Computational Thinking, Project-Based Learning, and AI Literacy

SPRINTA-BSD brings together three main orientations, namely computational thinking, project-based learning, and AI literacy. In terms of computational thinking, the model trains students to break down problems, recognize patterns, construct algorithms, test solutions, and debug. In terms of project-based learning, the model has students learn through projects that have outputs, success criteria, and opportunities for modification. In terms of AI literacy, the model opens space for students to understand how intelligent technologies work, are used, and need to be critically examined from an ethical perspective.

AI education research shows that AI education in the K-12 context frequently uses project-based learning, human-computer collaborative learning, game-based approaches, and constructivist methodologies (Yim & Su, 2025). Lee and Kwon (2024) also found that AI education covers the topics of basic AI concepts, AI applications, and ethical considerations. SPRINTA-BSD is relevant to these findings because it can accommodate projects such as a digital assistant, image classifier, AI image styler, or AI beat maker, while also providing space for discussion of data bias, responsible use of AI, and the impact of technology on work.

AI literacy cannot be understood merely as the ability to use AI tools. Long and Magerko (2020) stress the need for competencies to understand, use, evaluate, and critically consider AI. Celik (2023) shows that computational thinking is related to AI literacy because it helps individuals recognize, use, and evaluate AI-based technologies. Therefore, SPRINTA-BSD can serve as a bridge between coding, computational thinking, and AI literacy, especially if teachers connect projects with issues of digital ethics, bias, privacy, and responsible use of technology.

The ethical aspect needs to be strengthened because algorithms and AI systems can carry bias in educational contexts (Baker & Hawn, 2022). In SPRINTA-BSD, discussion of ethics can be included in the Resource Reading and Guided Exploration stage and the

Assessment, Reflection, and Advancement stage. Students are asked not only to make a project run, but also to answer reflective questions such as what data were used, who might be harmed by the system, how the output could be wrong, and how technology should be used responsibly.

The Relationship between SPRINTA-BSD and Naval Collaboration Flexible Learning

SPRINTA-BSD has a conceptual closeness to Naval Collaboration Flexible Learning (NCFL), developed by Dr. Adi Bandono. NCFL positions learning as a collaborative, flexible, and planned system that is tied to learners' real needs (Bandono, 2020b). In SPRINTA-BSD, this principle is seen in the integration of synchronous and asynchronous activities, independent and collaborative work, and the use of digital features to monitor the learning process.

The NCFL model is also understood as a learning technology innovation developed through a synthesis of learning theories, a systems approach, learners' needs, developments in science and technology, and institutional values (Bandono, 2020a). SPRINTA-BSD translates this idea into the coding context by connecting learning objectives, platform features, teacher activities, student activities, assessment, portfolios, and reflection.

The relationship between the two can be read through three points of convergence. First, both view learning as a system, not a single activity. Second, both place collaboration and flexibility as important principles. Third, both emphasize authentic learning experiences. If NCFL was developed in the context of maritime higher education (Bandono et al., 2020), then SPRINTA-BSD applies the collaborative-flexible spirit to project-based coding instruction in schools.

Assessment, Portfolios, and Reflection

The strength of SPRINTA-BSD lies in positioning assessment as part of the learning process. Assessment is not carried out only after the project is finished, but takes place from the time students read the objective, map the problem, explore resources, write code, test output, debug, save their work, and write reflections. In this way, teachers can assess students' development more fully.

The SPRINTA-BSD assessment rubric should include at least five components: (1) understanding of the problem and objective; (2) quality of logic and code structure; (3) debugging and revision ability; (4) creativity in project modification; and (5) reflection and communication of the process. These components are in line with the view that computational thinking requires assessment that measures not only correct answers but also strategies, processes, and the artifacts produced (Rao & Bhagat, 2024).

A digital portfolio is important evidence of learning development. Through the portfolio, students can show early versions of their work, error notes, improvements, modifications, and reflections. Teachers can also use the portfolio to give specific feedback, for example that the button logic is correct but the output display needs tidying, or that the input response works but the variety of functions can still be developed. Such feedback is more meaningful than general comments such as good or poor.

Reflection enables students to learn to explain their thinking process. Reflective questions can include: what problem did I solve, which part was the most difficult, what errors appeared, what strategy did I use to fix them, what feedback did I receive, and how will I develop the next project. In this way, SPRINTA-BSD positions students as learners who think, create, and dare to correct their mistakes.

Implementation Challenges and Mitigation Strategies

Implementing SPRINTA-BSD still requires readiness. The main challenges include infrastructure, teacher readiness, differences in students' learning pace, classroom culture toward errors, and the quality of assessment. For the model to be applied realistically, teachers need to prepare mitigation strategies as shown in Table 3.

Table 3. Challenges and Mitigation Strategies for Implementing SPRINTA-BSD

Challenges	Impact on Learning	Mitigation Strategy
Uneven access to devices and internet connection.	Students are late entering the classroom or cannot run the project.	Prepare access-check sessions, pair-work alternatives, offline materials, and a school device usage schedule.
Teachers are not yet familiar with the platform and the coding flow.	Instruction risks becoming a technical demonstration without pedagogical depth.	Conduct teacher training, provide step-by-step guides for each stage, a bank of guiding questions, and simulations before class.
Students' learning pace differs.	Students quickly become bored or fall behind in the early stages.	Use progress tracking for remediation, sandbox enrichment for fast students, and peer tutor groups.
Students fear making mistakes when they encounter errors.	Debugging is viewed as failure, not as a learning process.	Build a classroom culture in which errors are learning data; use debugging logs and solution discussions.
Assessment only evaluates the final product.	Thinking processes, revisions, and collaboration are invisible.	Use process rubrics, portfolios, feedback notes, and individual reflection.

These challenges do not diminish the value of SPRINTA-BSD, but show that the model needs to be applied with careful planning. Precisely because coding instruction has technical and pedagogical complexity, a model with a clear syntax, process assessment, and

reflection strategy is important for helping teachers manage digital learning in a more directed way.

Limitations of the Study and Further Research Agenda

This study has limitations. First, the research is still at the conceptual stage, so it has not tested the effectiveness of SPRINTA-BSD through classroom experiments. Second, expert validation of the syntax, learning tools, and assessment rubric has not been presented. Third, the platform analysis still focuses on the pedagogical suitability of the features, not on an in-depth evaluation of user experience.

Further research is recommended to use a research and development or design-based research design. Subsequent stages may include analysis of teacher and student needs, validation by subject-matter experts and instructional design experts, module readability testing, limited trials, model revision, and effectiveness testing on computational thinking, self-efficacy, creativity, and the quality of students' portfolios. Empirical testing also needs to consider variation in school contexts, device availability, initial digital literacy, and teacher readiness.

In addition, the development of SPRINTA-BSD assessment instruments needs to be carried out more systematically. The instruments may take the form of a project rubric, a coding activity observation sheet, a debugging log, a reflection guide, a presentation rubric, and a portfolio rubric. With more measurable instruments, SPRINTA-BSD can be evaluated not only as a conceptual idea but as a learning model that can be implemented and scientifically evaluated.

CONCLUSION

SPRINTA-BSD is a digital platform-based coding learning model that positions coding as a process of thinking, creating, collaborating, and reflecting on learning experiences. The model consists of six stages, namely Sign In and Stimulation, Project Orientation and Problem Mapping, Resource Reading and Guided Exploration, Interactive Coding and Idea Building, Networked Technology Action, and Assessment, Reflection and Advancement. These six stages form a learning cycle that connects technical readiness, problem orientation, resource literacy, coding practice, debugging, work modification, portfolios, feedback, and reflection.

The main contribution of SPRINTA-BSD lies in the integration of learning syntax, platform features, teacher activities, student activities, process assessment, and digital portfolios. The model strengthens coding instruction because it pursues not only the final product but also assesses the process of understanding problems, constructing logic, testing

output, correcting errors, receiving feedback, and developing work. SPRINTA-BSD is also aligned with the NCFL principles of collaboration, flexibility, authentic learning experiences, and a planned learning system.

Conceptually, SPRINTA-BSD deserves to be developed as a contemporary coding learning model to support computational thinking, AI literacy, creativity, independence, and a reflective culture among students. However, this article has not empirically proven the effectiveness of the model. Therefore, subsequent research needs to carry out expert validation, limited trials, development of assessment instruments, and testing of the model's impact on computational thinking ability and the quality of students' digital works.

BIBLIOGRAPHY

Arslan Namli, N., & Aybek, B. (2022). An investigation of the effect of block-based programming and unplugged coding activities on fifth graders' computational thinking skills, self-efficacy and academic performance. *Contemporary Educational Technology, 14*(1), Article ep341. https://doi.org/10.30935/cedtech/11477

Baker, R. S., & Hawn, A. (2022). Algorithmic bias in education. *International Journal of Artificial Intelligence in Education, 32*, 1052–1092. https://doi.org/10.1007/s40593-021-00285-9

Bandono, A. (2020a). The effect of the learning model Naval Collaboration Flexible Learning (NCFL) on the quality of education outcomes: An innovative approach to Islamic Religious Education in the Naval College of Technology. *Halaqa: Islamic Education Journal, 4*(2), 169–190. https://doi.org/10.21070/halaqa.v4i2.1088

Bandono, A. (2020b). *Model pembelajaran Naval Collaboration Flexible Learning (NCFL)* [Naval Collaboration Flexible Learning (NCFL) learning model]. STTAL.

Bandono, A., Rahman, A., & Suharyo, O. S. (2020). The application model of Naval Collaboration Flexible Learning (NCFL) in the Indonesian Naval Technology College: A case study. *International Journal of Progressive Sciences and Technologies, 23*(2), 498–505.

Bati, K. (2022). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies, 27*, 2059–2082. https://doi.org/10.1007/s10639-021-10700-2

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association* (pp. 1–25). Vancouver, Canada.

Camalia, A. R., & Rokhmaniyah. (2024). Analysis of the implementation of the Computational Thinking approach in Natural Science Learning in high elementary school classes. *Social, Humanities, and Educational Studies (SHES): Conference Series, 7*(3), 164–170. https://doi.org/10.20961/shes.v7i3.91541

Celik, I. (2023). Exploring the determinants of artificial intelligence (AI) literacy: Digital divide, computational thinking, cognitive absorption. *Telematics and Informatics, 83*, Article 102026. https://doi.org/10.1016/j.tele.2023.102026

Chai, C. S., Lin, P.-Y., Jong, M. S.-Y., Dai, Y., Chiu, T. K. F., & Qin, J. (2021). Perceptions of and behavioral intentions towards learning artificial intelligence in primary school students. *Educational Technology & Society, 24*(3), 89–101.

Chiu, T. K. F., Xia, Q., Zhou, X., Chai, C. S., & Cheng, M. (2023). Systematic literature review on opportunities, challenges, and future research recommendations of artificial intelligence in education. *Computers and Education: Artificial Intelligence, 4*, Article 100118. https://doi.org/10.1016/j.caeai.2022.100118

Christina, D., & Lukas, S. (2024). Efektivitas pembelajaran coding terhadap kemampuan computational thinking, problem solving, dan matematika siswa TK B TK XYZ Jakarta Utara [The effectiveness of coding learning on the computational thinking, problem solving, and mathematics abilities of kindergarten B students at TK XYZ, North Jakarta]. *EduTech: Jurnal Teknologi Pendidikan, 23*(3), 289–306. https://doi.org/10.17509/e.v23i3.73291

Darmayanti, I., Hermanto, N., & Subarkah, P. (2023). Pelatihan koding sebagai upaya meningkatkan computational thinking siswa [Coding training as an effort to improve students' computational thinking]. *SELAPARANG: Jurnal Pengabdian Masyarakat Berkemajuan, 7*(2), 833–838. https://doi.org/10.31764/jpmb.v7i2.14860

Fitriyah, Y., Wahyudin, Nurhayati, H., & Febrianti, T. S. (2024). Indonesian students' computational thinking performance based on level and gender. *International Journal of Pedagogy and Teacher Education, 8*(1).

Lee, S. J., & Kwon, K. (2024). A systematic review of AI education in K-12 classrooms from 2018 to 2023: Topics, strategies, and learning outcomes. *Computers and Education: Artificial Intelligence, 6*, Article 100211. https://doi.org/10.1016/j.caeai.2024.100211

Long, D., & Magerko, B. (2020). What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–16). ACM. https://doi.org/10.1145/3313831.3376727

Massaty, M. H., Fahrurozi, S. K., & Budiyanto, C. W. (2024). The role of AI in fostering computational thinking and self-efficacy in educational settings: A systematic review. *IJIE: Indonesian Journal of Informatics Education, 8*(1), 49. https://doi.org/10.20961/ijie.v8i1.89596

Mills, K. A., Cope, J., Scholes, L., & Rowe, L. (2025). Coding and computational thinking across the curriculum: A review of educational outcomes. *Review of Educational Research, 95*(3), 581–618. https://doi.org/10.3102/00346543241241327

Nur, M. A., & Nurhafidzah. (2025). Dampak dan tantangan pembelajaran coding bagi siswa sekolah dasar: A systematic literature review [The impact and challenges of coding learning for elementary school students: A systematic literature review]. *Jurnal Didaktika Pendidikan Dasar, 9*(3). https://doi.org/10.26811/didaktika.v9i3.2033

Rao, T. S. S., & Bhagat, K. K. (2024). Computational thinking for the digital age: A systematic review of tools, pedagogical strategies, and assessment practices. *Educational Technology Research and Development, 72*, 1893–1924. https://doi.org/10.1007/s11423-024-10364-y

Shin, N., Bowers, J., Krajcik, J., & Damelin, D. (2021). Promoting computational thinking through project-based learning. *Disciplinary and Interdisciplinary Science Education Research, 3*, Article 7. https://doi.org/10.1186/s43031-021-00033-y

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Tissenbaum, M., Sheldon, J., & Abelson, H. (2019). From computational thinking to computational action. *Communications of the ACM, 62*(3), 34–36. https://doi.org/10.1145/3265747

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (2019). Envisioning AI for K-12: What should every child know about AI? *Proceedings of the AAAI Conference on Artificial Intelligence, 33*(1), 9795–9799. https://doi.org/10.1609/aaai.v33i01.33019795

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of*

Science Education and Technology, 25, 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yim, I. H. Y., & Su, J. (2025). Artificial intelligence (AI) learning tools in K-12 education: A scoping review. *Journal of Computers in Education, 12*, 93–131. https://doi.org/10.1007/s40692-023-00304-9