

ENHANCING COMPUTATIONAL THINKING SKILLS AND SELF-EFFICACY OF PROSPECTIVE TEACHERS THROUGH SCRATCH-BASED PROJECT LEARNING

Firdaus Firdaus¹, Eli Trisnowati², Fatiatun Fatiatun¹, Filip Netušil³, Sri Jumini¹, Desty Putri Hanifah⁴, Aningsih¹

¹Physics Education Study Program, Faculty of Tarbiyah and Teacher Training, Universitas Sains Al-Qur'an, Wonosobo, Indonesia

²Natural Science Education Study Program, Faculty of Education and Teacher Training, Universitas Tidar, Magelang, Indonesia

³Asian studies, Palacký University in Olomouc, Olomouc, Czech Republic

⁴Madrasah Ibtidaiyah Teacher Education Study Program, Faculty of Tarbiyah and Teacher Training, Universitas Sains Al-Qur'an, Wonosobo, Indonesia

Article Info

Article history:

Received 02/07/2026

Accepted 09/10/2026

Published 16/10/2026

Keywords:

Computational thinking skills;

Project-based learning;

Prospective teachers;

Scratch;

Self-efficacy.

ABSTRACT

Computational thinking has become an essential competency for prospective teachers to support technology-integrated learning in the digital era. However, empirical studies integrating project-based learning with Scratch programming that simultaneously examine cognitive and affective outcomes in teacher education contexts remain limited. This research investigated the effectiveness of Scratch-integrated Project-Based Learning in enhancing prospective teachers' computational thinking skills and self-efficacy. A mixed-methods framework was adopted through a quasi-experimental design involving two groups, an experimental group and a control group. The participants consisted of 64 students enrolled in a teacher education program. Quantitative evidence was obtained through computational thinking assessments and self-efficacy questionnaires, whereas qualitative information was gathered through semi-structured interviews and analyzed using NVivo to provide deeper insights into students' learning processes and experiences. The quantitative dataset was examined using multivariate analysis of variance, normalized gain analysis, and effect size calculations. The findings demonstrated a significant multivariate difference associated with Scratch-integrated project-based learning, indicating its potential effectiveness in enhancing prospective teachers' computational thinking skills and self-efficacy. The experimental group demonstrated moderate learning gains and large effect sizes. Qualitative findings supported these results by showing increased engagement, collaboration, and confidence in computational problem-solving during project development. The study contributes theoretically by reinforcing the integration of constructivist learning principles with the development of computational thinking and the formation of self-efficacy.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Firdaus Firdaus

Physics Education Study Program, Faculty of Tarbiyah and Teacher Training, Universitas Sains Al-Qur'an, Wonosobo, Indonesia

Email: firdaus@unsiq.ac.id

1. INTRODUCTION

The growing influence of digital technologies across educational and professional settings has made computational thinking (CT) an increasingly important competency. Beyond its

relevance to computer science, CT enables learners to approach complex problems systematically through logical reasoning, algorithmic processes, and computational strategies. Its educational significance has therefore been recognized alongside foundational literacy and numeracy skills (Labusch et al., 2019). As computing continues to reshape academic disciplines and professional practices, students need opportunities to develop computational approaches to problem-solving from an early stage of education (Shute et al., 2017; Swaid, 2015). This expectation has particular implications for teacher education, as prospective teachers must develop their own computational competencies before they can meaningfully incorporate computational thinking into instructional design and facilitate its development among future learners.

Existing research on CT development has predominantly examined learning experiences among young children and primary school students (Bati, 2022; del Olmo-Muñoz et al., 2020; Lloyd & Chandra, 2020). Despite this growing body of evidence, translating CT concepts into effective classroom instruction remains challenging, particularly in determining appropriate pedagogical strategies for fostering students' computational competencies (Hsu et al., 2018). Programming offers a practical context for addressing this challenge by engaging learners in computational reasoning that extends beyond the technical process of writing code. Through programming-oriented projects, students can explore complex problems, formulate structured solutions, and apply computational principles to meaningful learning tasks (L. C. Zhang & Nouri, 2019).

Project-based learning (PjBL) provides a pedagogical framework for integrating programming activities into meaningful learning experiences, making it a relevant approach to CT development (Papavlasopoulou et al., 2019). By engaging learners in the design and completion of authentic projects, this approach creates opportunities to strengthen problem-solving abilities (Toker & Baturay, 2022) while promoting computational practices and perspectives (Saritepeci, 2020). Nevertheless, the pedagogical processes through which CT can be systematically developed remain insufficiently articulated in the literature. Hsu et al. (2018) identified limitations in existing CT instructional research, particularly regarding the explicit description of learning procedures and the availability of empirical evidence. These limitations highlight the need for instructional approaches that translate computational concepts into structured learning activities. Accordingly, the present study investigates the integration of PjBL and Scratch programming as an instructional strategy for developing computational thinking skills among prospective teachers.

Grounded in constructivist principles, PjBL emphasizes active knowledge construction through sustained inquiry, exploration, and collaborative problem-solving. Rather than relying primarily on direct instruction, learners engage in meaningful tasks that require them to investigate problems, evaluate alternative solutions, and develop tangible outcomes. Shin et al. (2021) outlined a pedagogical framework connecting PjBL with CT through four interrelated components. The first involves establishing learning objectives that integrate scientific understanding with computational practices. The second introduces a guiding question to stimulate inquiry and encourage computational reasoning. The third engages learners in investigating the question through scientific and computational activities that connect conceptual knowledge with real-world contexts. Finally, students apply their understanding to develop computational products, allowing them to demonstrate and refine their knowledge through active participation. Together, these components provide a structured foundation for incorporating CT into project-oriented learning environments.

Scratch, a visual programming environment, provides a practical platform for implementing PjBL by enabling learners to design interactive stories, animations, and games through block-based programming. Its accessible interface allows students to engage with computational concepts while minimizing the technical difficulties commonly associated with text-based programming. Incorporating Scratch into learning activities can support the development of higher-order thinking skills, particularly problem-solving, creativity, logical reasoning, and computational thinking (Akcaoglu, 2014). Within this context, integrating Scratch with PjBL offers opportunities for prospective teachers to develop computational competencies through the design, implementation, and refinement of digital projects. Drawing on the framework of Shin et al. (2021), this study examines five dimensions of computational thinking: (CT1) formulating problems in computationally solvable terms, (CT2) developing computational artifacts through algorithmic reasoning, (CT3) collecting, organizing, and interpreting data, (CT4) evaluating program functionality through testing and debugging, and (CT5) progressively refining solutions through iterative improvement.

Beyond fostering computational thinking skills, Scratch-based programming activities may also contribute to learners' self-efficacy in performing computational tasks. Yukselturk & Altıok (2017) reported that engagement with Scratch programming positively influenced prospective teachers' confidence in their programming capabilities. From an educational perspective, self-efficacy reflects individuals' beliefs about their ability to acquire knowledge and successfully accomplish tasks within a particular domain (Martin & Rimm-Kaufman, 2015). Such beliefs play an important role in shaping learners' motivation, cognitive engagement, and willingness to participate in challenging learning activities. In programming contexts, students with stronger self-efficacy may demonstrate greater persistence when encountering computational difficulties, increased confidence in exploring alternative solutions, and a greater willingness to engage in complex problem-solving tasks. Accordingly, examining self-efficacy alongside computational thinking skills provides a broader perspective on the potential cognitive and affective benefits of integrating Scratch programming into project-based learning among prospective teachers.

Despite growing scholarly interest in Scratch-supported CT instruction, several areas warrant further investigation, particularly within teacher education. Previous studies have frequently examined the development of computational competencies among primary and secondary school students, predominantly in science, technology, engineering, and mathematics contexts. However, the preparation of prospective teachers to integrate computational approaches into their future instructional practices requires greater empirical attention. Furthermore, although Scratch programming and PjBL have each been investigated as instructional approaches, further evidence is needed to clarify their combined contribution to computational thinking development in teacher preparation programs. Existing research has also placed considerable emphasis on cognitive learning outcomes (Augusto Hernández Suárez et al., 2022; Tsarava et al., 2019), leaving opportunities to examine affective factors, particularly learners' self-efficacy in addressing computational challenges. Investigating these dimensions together may provide a more comprehensive understanding of how computational learning experiences shape both problem-solving competencies and confidence in applying them. To address these research needs, the present study evaluates the integration of Scratch programming and PjBL in prospective teacher education, with particular attention to changes in computational thinking skills and self-efficacy.

Building on the identified research gaps, this study seeks to advance the understanding of

CT development in teacher education through three complementary contributions. First, it provides empirical evidence from prospective teachers, extending the educational context in which computational thinking interventions are examined and addressing the need to better understand their preparation for technology-integrated instruction. Second, this research contributes to educational technology literature by examining the combination of PjBL and Scratch programming as an integrated instructional framework for developing both cognitive and affective learning outcomes. Through the simultaneous assessment of computational thinking skills and self-efficacy, the study provides empirical evidence regarding how structured programming projects may facilitate students' computational development and confidence in engaging with computational tasks. Third, the study adopts a mixed-methods design that integrates quantitative comparisons through a quasi-experimental approach with qualitative interpretations derived from participants' learning experiences, enabling a more comprehensive understanding of the learning process and outcomes. This methodological integration enables quantitative differences in learning outcomes to be interpreted alongside students' perceptions of engagement, problem-solving, and confidence during project development. Collectively, these contributions provide an empirical basis for considering Scratch-integrated PjBL as an instructional approach for strengthening prospective teachers' computational competencies and confidence in technology-supported learning environments.

In response to the identified research gaps, this study aims to evaluate the effectiveness of PjBL integrated with Scratch programming in enhancing prospective teachers' CT skills and self-efficacy. Specifically, the study seeks to (1) determine the extent to which Scratch-integrated PjBL improves computational thinking performance compared with conventional Scratch-based instruction; (2) examine its effectiveness in strengthening prospective teachers' self-efficacy in computational problem-solving; and (3) assess the combined effect of the instructional approach on computational thinking skills and self-efficacy through multivariate analysis. By addressing these objectives, the study seeks to provide empirical evidence regarding the potential of Scratch-integrated PjBL to support both cognitive and affective learning outcomes in teacher education.

2. METHOD

This study employed a mixed-methods approach incorporating a quasi-experimental design with experimental and control groups, as illustrated in Figure 1. Quantitative data were obtained through assessments of computational thinking skills and self-efficacy, while qualitative insights were gathered through observations, product evaluations, and semi-structured interviews. Both groups participated in Scratch-based programming activities but received different instructional treatments. The experimental group engaged in project-based learning (PjBL) integrated with Scratch, whereas the control group followed conventional teacher-centred instruction using the same programming platform. The intervention was conducted over ten weeks, including pretest and posttest assessments, with each group attending one 100-minute session per week. To maintain consistency across instructional conditions, both groups were taught by the same instructor, assisted by a classroom facilitator, and provided with equivalent learning materials and instructional time.

Prior to the instructional intervention, participants in both the experimental and control groups completed pretest assessments to establish their baseline computational thinking skills and

self-efficacy in performing computational tasks. Following eight weeks of instructional activities, both groups underwent posttest assessments to determine changes in these two learning outcomes. The Scratch projects developed by participants during the intervention were evaluated using Dr. Scratch (<http://www.drscratch.org/>), an online platform for analyzing computational characteristics of Scratch programs. To complement the quantitative findings, qualitative data were obtained through semi-structured group interviews conducted with participants in the experimental group after the intervention. These interviews provided insights into students' learning experiences and perceptions of computational problem-solving activities.

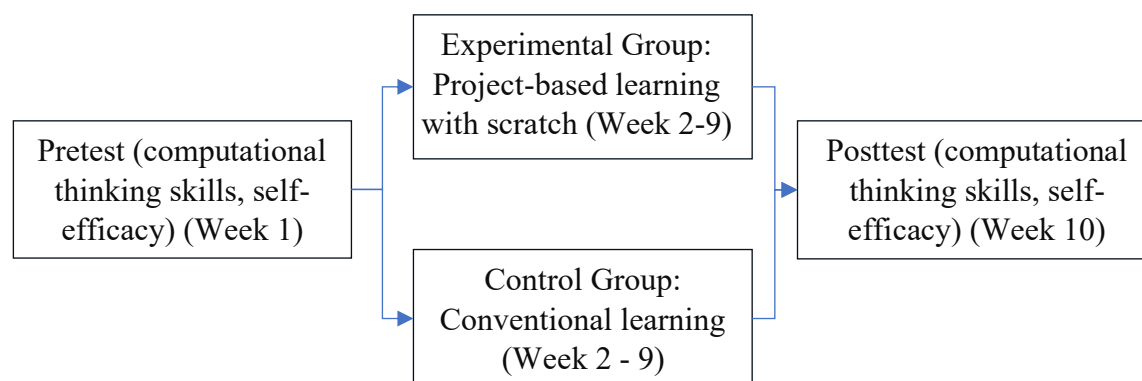


Figure 1. Research design

The study involved 64 prospective elementary school teachers enrolled at Universitas Sains Al-Qur'an, Indonesia. Participants were organized into two existing classes, comprising 32 students in the experimental group and 32 in the control group. The instructional intervention was implemented within the Learning Media and Technology course, where Scratch programming was incorporated into the learning activities. The demographic characteristics of participants in both groups are presented in Table 1.

Table 1. Demographic data of the experimental and control groups

Groups	Number of Students	Gender	
		Male	Female
Experimental Group	32	19	13
Control Group	32	20	12

The experimental class will engage in project-based learning using Scratch programming. The learning steps to be implemented are presented in Table 2.

Table 2. Steps for Scratch-Based Project Learning Activities

Project Learning Steps	Main Activities of Project-Based Learning
1. Focus on learning objectives	- Developing performance-based learning objectives by integrating theory with practice
2. Driver questions	- Using questions to guide student activities
3. Participate in simulations	- Define the problem or phenomenon to be projected

-
- | | |
|--|---|
| 4. Create a product through collaborative activities | <ul style="list-style-type: none"> - Define the boundaries of the project to be developed - Creating a computational model - Run a simulation - Create graphs based on simulation data - Designing collaborative learning activities for students to create products - Work in pairs and evaluate the products - Communicating products to the classroom |
|--|---|
-

The control group received conventional teacher-centred instruction incorporating Scratch programming. Learning activities were primarily facilitated through instructor-led explanations, demonstrations, and step-by-step guidance, with students following predetermined instructional procedures. Although Scratch was used as the programming platform, the learning process emphasized the acquisition and application of demonstrated procedures rather than independent project exploration and collaborative problem-solving. This instructional arrangement served as a comparison condition for evaluating the effectiveness of Scratch-integrated project-based learning implemented in the experimental group.

CT abilities among prospective teachers were measured through an adapted assessment instrument originally developed by Román-González et al. (2017). The assessment framework incorporated five CT components based on Shin et al. (2021), including the ability to transform complex problems into computationally solvable forms (CT1), develop computational artifacts using algorithmic reasoning (CT2), generate, organize, and interpret data representations (CT3), identify and correct errors through testing and debugging processes (CT4), and improve computational solutions through repeated refinement cycles (CT5). As part of the assessment process, participants created Scratch-based projects representing their computational solutions, which were analyzed using Dr. Scratch (<http://www.drscratch.org/>), an automated web-based tool designed to evaluate computational features embedded in Scratch programs (Moreno-León et al., 2015). The content validity analysis demonstrated satisfactory expert agreement, with Aiken's V coefficients ranging from 0.83 to 0.96 and an overall mean value of 0.90. Furthermore, the reliability evaluation showed a Cronbach's alpha coefficient of 0.91, indicating a high level of internal consistency among the assessment items. These results confirm that the instrument provides adequate evidence of content validity and reliability for evaluating computational thinking skills in the context of prospective teacher education.

The assessment of prospective teachers' self-efficacy related to CT was conducted using a modified questionnaire based on the instrument developed by Korkmaz & Bai (2019). The questionnaire consisted of 22 statements measured through a five-level Likert response format. The instrument evaluated five self-efficacy dimensions associated with computational thinking activities. In addition to quantitative measurement, qualitative evidence was collected through semi-structured group interviews involving participants from the experimental group after the completion of the intervention. The interview protocol was adapted from Grover & Pea (2016) and focused on students' experiences in approaching computational problems, challenges encountered during programming activities, and their future learning interests in programming. All interview recordings were converted into written transcripts and analyzed through a systematic qualitative procedure involving transcript review, data coding, and identification of meaningful

themes. The validity analysis of the self-efficacy questionnaire produced Aiken's V coefficients between 0.82 and 0.94, with an average value of 0.89, demonstrating strong expert agreement regarding the relevance of the items. Furthermore, the reliability test resulted in a Cronbach's alpha value of 0.87, indicating satisfactory internal consistency. These statistical evaluations confirm that the instrument was appropriate for examining prospective teachers' self-efficacy in computational thinking within this research context.

The effectiveness of Scratch-integrated project-based learning was examined through a combination of descriptive and inferential statistical procedures. Prior to hypothesis testing, preliminary diagnostic analyses were conducted to verify whether the dataset satisfied the assumptions required for parametric analysis, including normal distribution and homogeneity of variance. Since the study involved two outcome variables, namely computational thinking skills and self-efficacy, a multivariate analysis of variance (MANOVA) was performed to determine whether the experimental and control groups differed in their overall post-intervention performance. This multivariate approach enabled the simultaneous evaluation of cognitive and affective outcomes resulting from the different instructional conditions. Furthermore, learning improvement was assessed by calculating gain scores derived from the difference between pretest and posttest results. The normalized gain (N-gain) index was subsequently applied to estimate the degree of improvement achieved by each group and to classify the magnitude of learning gains across the measured outcomes.

Beyond evaluating statistical significance, the practical relevance of the instructional effect was examined through Cohen's d effect size analysis. This measure was used to quantify the standardized difference in learning outcomes between the experimental and control groups. The magnitude of the effect was interpreted according to established criteria, in which values of 0.20, 0.50, and 0.80 indicate small, moderate, and large effects, respectively (Sullivan & Feinn, 2013). Additionally, qualitative data obtained from semi-structured student interviews were examined through thematic analysis supported by NVivo software. The analysis focused on identifying recurring patterns in participants' learning experiences, computational problem-solving processes, and perceived self-efficacy. The resulting themes were used to contextualize and enrich the interpretation of the quantitative findings.

3. RESULTS AND DISCUSSION

3.1. Result

Prior to conducting the MANOVA, preliminary assumption checks were carried out to ensure that the data fulfilled the statistical requirements for multivariate analysis. These examinations included evaluations of distributional normality, equality of variance–covariance matrices, linear associations among dependent variables, and potential multicollinearity issues. The results of these diagnostic assessments confirmed that the dataset satisfied the assumptions necessary for proceeding with multivariate testing. Subsequently, MANOVA was applied to determine whether the instructional treatments resulted in significant differences between the experimental and control groups when computational thinking skills and self-efficacy were considered simultaneously as dependent variables.

The MANOVA results revealed a statistically significant multivariate difference between

the experimental and control groups across the combined outcomes of computational thinking skills and self-efficacy (Wilks' $\Lambda = 0.721$, $p < 0.001$, $\eta^2 = 0.279$). This finding suggests that the instructional approaches were associated with significantly different patterns of cognitive and affective learning outcomes, providing evidence of a multivariate effect of Scratch-integrated project-based learning.

The univariate follow-up analysis indicated a significant difference in computational thinking outcomes between the two instructional groups ($p = 0.001$, $\eta^2 = 0.198$). Participants in the experimental group obtained a higher mean posttest score ($M = 78.2$) compared with those in the control group ($M = 69.9$), resulting in an observed difference of 8.3 points. This result indicates that prospective teachers involved in Scratch-integrated project-based learning achieved stronger computational thinking performance than participants who experienced conventional Scratch-based instruction. The detailed descriptive results for computational thinking scores across both groups are summarized in Table 3.

Table 3. Computational Thinking Ability Scores

Groups	N	Pretest Mean	Posttest Mean
Experiment	32	65.5	78.2
Control	32	65.2	69.9

The univariate analysis conducted after the MANOVA revealed a significant difference in self-efficacy outcomes between participants in the experimental and control groups ($p = 0.002$, $\eta^2 = 0.157$). The experimental group demonstrated a higher level of self-efficacy at the posttest stage, with a mean score of 83.4, compared with 74.2 for the control group. The difference between the two groups corresponded to a mean increase of 9.2 points, indicating a more favorable self-efficacy outcome among students who participated in Scratch-integrated project-based learning. This result indicates that prospective teachers exposed to Scratch-integrated project-based learning demonstrated greater self-efficacy in computational problem-solving than their counterparts who received conventional Scratch-based instruction. The descriptive statistics for self-efficacy scores in both groups are presented in Table 4.

Table 4. Self-Efficacy Scores

Groups	N	Pretest Mean	Posttest Mean
Experiment	32	70.2	83.4
Control	32	70.2	74.2

The magnitude of improvement in students' learning outcomes was further examined through normalized gain (N-gain) analysis. The calculation was based on the differences between pretest and posttest scores obtained by participants in both the experimental and control groups. This analysis enabled a comparative evaluation of the relative improvement achieved under each instructional condition for both computational thinking skills and self-efficacy.

Table 5. N-Gain Results of Computational Thinking Skills

Groups	N	N-Gain	Category
Experiment	32	0.37	Medium
Control	32	0.14	Low

As presented in Table 5, the experimental group demonstrated a moderate improvement in computational thinking skills, with a mean normalized gain of 0.37. In contrast, the control group recorded a substantially lower N-gain of 0.14, corresponding to the low improvement category. These results indicate greater relative learning gains among participants receiving Scratch-integrated project-based learning than among those undergoing conventional Scratch-based instruction.

Table 6. N-Gain Results of Self-Efficacy

Groups	N	N-Gain	Category
Experiment	32	0.45	Medium
Control	32	0.12	Low

The N-gain analysis shows that students in the experimental class experienced a moderate increase in self-efficacy ($g = 0.45$), whereas the control class demonstrated only a low level of improvement ($g = 0.12$). These findings indicate that project-based learning integrated with Scratch programming produces greater learning gains compared to conventional instruction. The medium N-gain achieved by the experimental group suggests that active project engagement supports meaningful improvement in both computational thinking performance and students' confidence in computational problem-solving. In contrast, the low gain observed in the control group implies that conventional Scratch learning provides limited opportunities for sustained cognitive and affective development.

The practical magnitude of the intervention effects was evaluated through Cohen's d effect size analysis. The effect size estimation was derived from the standardized difference between the posttest mean scores of the experimental and control groups, with the pooled standard deviation used as the reference measure of score variability. The analysis indicated substantial effect sizes for Scratch-integrated project-based learning, yielding Cohen's d values of 0.82 for computational thinking skills and 0.88 for self-efficacy. These results suggest that the differences observed between the two groups were not only statistically meaningful but also demonstrated considerable educational relevance, highlighting the potential contribution of project-based computational learning in strengthening both cognitive performance and learners' confidence in computational activities.

In addition to the quantitative findings, qualitative data from student interviews were used to explain and enrich the statistical results obtained from the experimental intervention. Interview data related to students' learning experiences and self-efficacy in computational thinking were analyzed using NVivo. The analysis indicated that students perceived Scratch-based project activities as engaging and supportive in facilitating conceptual understanding and problem-solving processes. Participants reported increased confidence in solving programming problems, greater willingness to explore alternative solutions, and improved collaboration during project development. These qualitative insights complement the quantitative findings by explaining how

project-based learning experiences contributed to significant improvements in computational thinking performance and self-efficacy. The convergence between quantitative and qualitative results confirms the effectiveness of Scratch-based project learning from both performance and experiential perspectives.

3.2. Discussion

The results of this investigation indicate that the implementation of Scratch-integrated project-based learning was associated with significant improvements in prospective teachers' computational thinking abilities and self-efficacy. From a theoretical perspective, these findings can be explained through constructivist learning theory, which emphasizes that meaningful learning develops when learners actively build their own understanding through interaction, exploration, and authentic experiences rather than merely receiving knowledge transmitted by instructors (Vygotsky, 1978). In a constructivist learning environment, students are encouraged to interact with problems (Çibukçiu, 2025; Firdaus, 2017; Savaengkan & Chaijaroen, 2025), explore alternative solutions (Bou Saad et al., 2025), and reflect on their learning processes (Del Valle & Guhao Jr., 2024; Qu & Chaijaroen, 2025; Subba et al., 2025), thereby fostering deeper conceptual understanding. The implementation of project-based learning in this study created authentic, meaningful learning contexts in which prospective teachers collaboratively designed, developed, and refined Scratch-based projects. Such activities enabled learners to engage in experiential learning cycles involving investigation, experimentation, evaluation, and revision, all of which are essential components in developing computational thinking skills.

Through project development assignments, students are introduced to theoretical explanations of programming concepts, and are simultaneously challenged to apply logical reasoning and systematic problem-solving strategies in real situations. The process of identifying problems, decomposing tasks into manageable steps, constructing algorithms, testing program functionality, and debugging errors reflects core computational thinking practices that emerge naturally during project completion. Moreover, reflective activities embedded in project-based learning encouraged students to evaluate their learning progress and reconsider alternative computational strategies, thereby strengthening higher-order thinking skills (Loyens et al., 2023; W. Zhang et al., 2024). This active engagement contrasts with conventional instructional approaches that often emphasize procedural knowledge acquisition at the expense of sufficient opportunities for exploration and creativity. Consequently, the learning experience became more student-centred, promoting autonomy, collaboration, and sustained cognitive engagement, which collectively contributed to the observed enhancement in both computational thinking performance and students' confidence in their ability to solve computational problems.

From the perspective of computational thinking theory, skills such as decomposition, algorithmic thinking, debugging, and iterative improvement are more effectively developed through active creation rather than passive instruction (Kwon & Cheon, 2019; MacCallum, 2025). The significant improvement observed in the experimental group indicates that Scratch-based project activities successfully facilitated students' engagement in structured problem-solving processes, including designing algorithms, testing programs, and refining solutions. The visual, block-based programming environment of Scratch reduces programming complexity and minimizes syntax-related difficulties, allowing learners to focus more on logical reasoning and

computational problem-solving. This finding supports theoretical assumptions that block-based programming environments function as cognitive scaffolding, helping novice programmers gradually develop computational reasoning (Noordin, 2025; Pršala, 2024; Tikva & Tambouris, 2023) and higher-order thinking skills (Changsri et al., 2024).

The findings further suggest that computational thinking development should not be viewed solely as a cognitive process but also as one involving affective factors, particularly learners' self-efficacy. This interpretation is consistent with social cognitive theory, which explains that individuals' beliefs about their capabilities are shaped through meaningful learning experiences, social interactions, and feedback received during task engagement. In the context of Scratch-based project learning, opportunities to design computational products, collaborate with peers, overcome programming challenges, and receive guidance may contribute to strengthening learners' confidence in performing computational tasks (Aikens & Kulacki, 2023; Woreta et al., 2025). In this study, students in the experimental group were actively involved in designing, testing, and improving Scratch-based projects, thereby experiencing successful task completion. These repeated mastery experiences strengthened students' confidence in their ability to solve computational problems, which explains the significant improvement in self-efficacy compared to students who used conventional learning approaches. In addition, collaborative project activities encouraged peer interaction and knowledge sharing, enabling students to learn from others' strategies and collectively overcome programming difficulties. Continuous instructor feedback and project evaluations further supported students in recognizing their progress, fostering motivation and persistence when facing complex computational tasks (Yu et al., 2025). As a result, project-based learning enhanced students' technical competence while also promoting positive learning attitudes and confidence essential for sustained engagement in computational problem-solving.

The implications of these findings are highly relevant for teacher education programs in preparing future educators to meet the demands of 21st-century learning environments. Prospective teachers require adequate computational competencies to effectively implement technology-integrated pedagogy that promotes critical thinking, creativity, collaboration, and digital literacy among students. The results of this study indicate that integrating Scratch-based project learning into teacher preparation curricula can simultaneously foster pedagogical creativity, enhance problem-solving abilities, and strengthen technological confidence among prospective teachers. Through hands-on project experiences, prospective teachers gain a practical understanding of how computational concepts can be applied in meaningful learning activities suitable for elementary education contexts. Furthermore, teachers who possess strong computational thinking skills and high self-efficacy are more ready to adopt innovative instructional strategies, design interactive digital learning media, and integrate coding or computational problem-solving activities into classroom practice. Consequently, embedding project-based programming experiences in teacher education programs can help develop adaptive, confident, and technologically competent teachers who can facilitate future-oriented learning.

At the institutional and policy levels, these findings support the integration of computational thinking as a core competency within prospective teacher education programs. Higher education institutions are encouraged to redesign curricula by embedding computational thinking and programming experiences into learning activities that emphasize active creation rather than mere technology usage. Educational policymakers should consider shifting from

technology-use training models to computational, creation-based learning approaches that prioritize design processes, creativity, and problem-solving skills. This transformation enables prospective teachers to develop deeper technological understanding and pedagogical readiness in applying digital tools meaningfully in classroom instruction. Strengthening computational thinking in teacher education is essential to prepare future educators to promote digital literacy, logical reasoning, and computational problem-solving among students, thereby supporting learners equipped for increasingly technology-driven educational environments.

Although the findings provide encouraging evidence regarding the potential of Scratch-integrated project-based learning, several limitations need to be considered. First, the relatively brief implementation period may not fully capture the long-term persistence of computational thinking abilities and self-efficacy after the intervention has ended. Second, the participants were recruited from only one higher education institution, which may restrict the extent to which the findings can be generalized to other teacher education contexts with different characteristics. Future studies are encouraged to adopt longitudinal research designs to investigate the sustainability of learning outcomes over extended periods, involve participants from diverse institutions to strengthen external validity, and explore interdisciplinary instructional models, including STEM- or STEAM-oriented approaches. These directions may contribute to a more comprehensive understanding of how project-based computational learning can be implemented sustainably and expanded across various teacher education environments.

4. CONCLUSION

This study provides empirical evidence that integrating project-based learning PjBL with Scratch programming can support the development of computational thinking skills and self-efficacy among prospective teachers. Compared with conventional Scratch-based instruction, the project-based approach was associated with significantly higher computational thinking performance and self-efficacy, supported by moderate normalized learning gains and large effect sizes. These findings suggest that engaging prospective teachers in the design, testing, and refinement of computational projects offers meaningful opportunities to strengthen their problem-solving competencies and confidence in undertaking programming-related tasks. The study contributes to teacher education research by highlighting the value of examining cognitive and affective learning outcomes within an integrated computational learning environment. From a practical perspective, incorporating Scratch-based projects into teacher preparation programs represents a promising pedagogical strategy for developing computational competencies and supporting future technology-integrated instructional practices. Further research involving broader participant populations and longitudinal designs is needed to establish the generalizability and sustainability of these outcomes.

REFERENCES

- Aikens, M. L., & Kulacki, A. R. (2023). Identifying Group Work Experiences That Increase Students' Self-Efficacy for Quantitative Biology Tasks. *CBE Life Sciences Education*, 22(2), 1–20. <https://doi.org/10.1187/cbe.22-04-0076>
- Akcaoglu, M. (2014). Learning problem-solving through making games at the game design and

- learning summer program. *Educational Technology Research and Development*, 62(5), 583–600. <https://doi.org/10.1007/s11423-014-9347-4>
- Augusto Hernández Suárez, C., Aloiso Gamboa Suárez, A., & Prada Núñez, R. (2022). Computational Thinking And Development Of Cognitive Skills Mediated By Unplugged Activities. *Journal of Language and Linguistic Studies*, 18(2), 1110–1122. www.jlls.org<https://orcid.org/0000-0002-7974-5560><https://orcid.org/0000-0001-9755-6408><https://orcid.org/0000-0001-6145-1786>
- Bati, K. (2022). A systematic literature review regarding computational thinking and programming in early childhood education. *Education and Information Technologies*, 27(2), 2059–2082. <https://doi.org/10.1007/s10639-021-10700-2>
- Bou Saad, R., Garcia, A. L., & Garcia, J. M. C. (2025). Mapping Constructivist Active Learning for STEM: Toward Sustainable Education. *Sustainability (Switzerland)*, 17(13), 1–20. <https://doi.org/10.3390/su17136225>
- Changsri, N., Inprasitha, M., Tang, K., Boonsena, N., & Pattanajak, A. (2024). The use of pattern blocks in developing students' higher-order thinking skills. *International Journal for Lesson and Learning Studies*, 13(4), 246–264. <https://doi.org/10.1108/IJLLS-04-2023-0038>
- Çibukçiu, B. (2025). The impact of constructivist methods on students' mathematical problem-solving. *Discover Education*, 4(83). <https://doi.org/10.1007/s44217-025-00475-w>
- del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of Primary Education. *Computers and Education*, 150(January). <https://doi.org/10.1016/j.compedu.2020.103832>
- Del Valle, J. M., & Guhao Jr., E. S. (2024). Constructivist Learning Environment Management, Metacognitive Awareness and Epistemological Beliefs: a Structural Equation Model on Academic Success of Students in Public Senior High Schools. *European Journal of Education Studies*, 11(7), 219–248. <https://doi.org/10.46827/ejes.v11i7.5424>
- Firdaus, F. (2017). Pengembangan Media Pembelajaran Bervisi Sets Berbantuan Komputer Untuk Meningkatkan Keterampilan Pemecahan Masalah. *Indonesian Journal of Science and Education Indonesian*, 1(1), 17–29.
- Grover, S., & Pea, R. (2016). Designing a blended, middle school computer science course for deeper learning: A design-based research approach. *Proceedings of International Conference of the Learning Sciences, ICLS*, 1(June), 695–702.
- Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and Education*, 126(September 2017), 296–310. <https://doi.org/10.1016/j.compedu.2018.07.004>
- Korkmaz, Ö., & Bai, X. (2019). Adapting computational thinking scale (CTS) for chinese high school students and their thinking scale skills level. *Participatory Educational Research*, 6(1), 10–26. <https://doi.org/10.17275/per.19.2.6.1>
- Kwon, K., & Cheon, J. (2019). Exploring problem decomposition and program development through block-based programs. *International Journal of Computer Science Education in Schools*, 3(1), 3–16. <https://doi.org/10.21585/ijcses.v3i1.54>
- Labusch, A., Eickelmann, B., & Vennemann, M. (2019). Computational thinking processes and their congruence with problem-solving and information processing. In *Computational Thinking Education*. https://doi.org/10.1007/978-981-13-6528-7_5

- Lloyd, M., & Chandra, V. (2020). Teaching coding and computational thinking in primary classrooms: perceptions of Australian preservice teachers. *Curriculum Perspectives*, 40(2), 189–201. <https://doi.org/10.1007/s41297-020-00117-1>
- Loyens, S. M. M., van Meerten, J. E., Schaap, L., & Wijnia, L. (2023). Situating Higher-Order, Critical, and Critical-Analytic Thinking in Problem- and Project-Based Learning Environments: A Systematic Review. *Educational Psychology Review*, 35(39), 1–44. <https://doi.org/10.1007/s10648-023-09757-x>
- MacCallum, K. (2025). Integrating computational thinking through digital creation: The (TPAC)2K model. *Teaching and Teacher Education*, 161(April), 105056. <https://doi.org/10.1016/j.tate.2025.105056>
- Martin, D. P., & Rimm-Kaufman, S. E. (2015). Do student self-efficacy and teacher-student interaction quality contribute to emotional and social engagement in fifth grade math? *Journal of School Psychology*, 53(5), 359–373. <https://doi.org/10.1016/j.jsp.2015.07.001>
- Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia (RED)*, 46. <https://doi.org/10.6018/red/46/10>
- Noordin, N. H. (2025). Computational Thinking Through Scaffolded Game Development Activities: A Study with Graphical Programming. *European Journal of Educational Research*, 14(4), 1137–1149. <https://doi.org/10.12973/eu-jer.14.4.1137>
- Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, 99(7491), 415–427. <https://doi.org/10.1016/j.chb.2019.01.008>
- Pršala, J. (2024). The effect of block-based programming activities on the computational thinking skills of pre-service primary school teachers. *Journal of Pedagogical Sociology and Psychology*, 6(3), 1–9. <https://doi.org/10.33902/jpsp.202426267>
- Qu, X., & Chaijaroen, S. (2025). A study on the design of constructivist learning environment model based on metacognitive regulation enhancement. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 124(December 2024), 659–680. <https://doi.org/10.61091/jcmcc124-44>
- Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, 72, 678–691. <https://doi.org/10.1016/j.chb.2016.08.047>
- Saritepeci, M. (2020). Developing computational thinking skills of high school students: Design-based learning activities and programming tasks. *Asia-Pacific Education Researcher*, 29(1), 35–54. <https://doi.org/10.1007/s40299-019-00480-2>
- Savaengkan, O., & Chaijaroen, S. (2025). Advancing Skills-Based Education: Designing and Validating a Constructivist Learning Environment Model to Foster Students' Ill-Structured Problem-Solving and Industrial Competencies. *Journal of Education and Learning*, 15(2), 195–212. <https://doi.org/10.5539/jel.v15n2p195>
- Shin, N., Bowers, J., Krajcik, J., & Damelin, D. (2021). Promoting computational thinking through project-based learning. *Disciplinary and Interdisciplinary Science Education Research*, 3(1). <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>
- Subba, B. H., Chanunan, S., & Poonpaiboonpipat, W. (2025). A proposed constructivism-based instructional model to enhance metacognition and mathematical problem-solving skills in Bhutanese grade nine students. *Journal on Mathematics Education*, 16(1), 51–72. <https://doi.org/10.22342/jme.v16i1.pp51-72>
- Sullivan, G. M., & Feinn, R. (2013). Using Effect Size—or Why the P Value Is Not Enough. *Journal of Graduate Medical Education*, 4(3), 279–282. <https://doi.org/10.4300/jgme-d-12-00156.1>
- Swaid, S. I. (2015). Bringing Computational Thinking to STEM Education. *Procedia Manufacturing*, 3(Ahfe), 3657–3662. <https://doi.org/10.1016/j.promfg.2015.07.761>
- Tikva, C., & Tambouris, E. (2023). The effect of scaffolding programming games and attitudes towards programming on the development of Computational Thinking. *Education and Information Technologies*, 28(6), 6845–6867. <https://doi.org/10.1007/s10639-022-11465-y>
- Toker, S., & Baturay, M. H. (2022). Developing disposition to critical thinking and problem-solving perception in instructional design projects for producing digital materials. *International Journal of Technology and Design Education*, 32(2), 1267–1292. <https://doi.org/10.1007/s10798-020-09646-2>
- Tsarava, K., Leifheit, L., Ninaus, M., Román-González, M., Butz, M. V., Golle, J., Trautwein, U., & Moeller, K. (2019). Cognitive correlates of computational thinking: Evaluation of a blended unplugged/Plugged-in course. *Proceedings of the 14th Workshop in Primary and Secondary Computing Education*, October 2019. <https://doi.org/10.1145/3361721.3361729>
- Vygotsky, L. (1978). Interaction between learning and development. In *Mind and society* (pp. 79–91). Harvard University Press. [https://doi.org/10.1016/S0006-3495\(96\)79572-3](https://doi.org/10.1016/S0006-3495(96)79572-3)
- Woreta, G. T., Zewude, G. T., & Józsa, K. (2025). The Mediating Role of Self-Efficacy and Outcome Expectations in the Relationship Between Peer Context and Academic Engagement: A Social Cognitive Theory Perspective. *Behavioral Sciences*, 15(5), 1–24. <https://doi.org/10.3390/bs15050681>
- Yu, H., Chen, Y., & Ismail, I. M. bin. (2025). From scaffolding to success: How instructor pedagogical support and collaborative classroom interaction drive scholastic motivation in programming education. *Acta Psychologica*, 259, 105289. <https://doi.org/10.1016/j.actpsy.2025.105289>
- Yukselturk, E., & Altiok, S. (2017). An investigation of the effects of programming with Scratch on the preservice IT teachers' self-efficacy perceptions and attitudes towards computer programming. *British Journal of Educational Technology*, 48(3), 789–801. <https://doi.org/10.1111/bjet.12453>
- Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers and Education*, 141(June), 103607. <https://doi.org/10.1016/j.compedu.2019.103607>
- Zhang, W., Guan, Y., & Hu, Z. (2024). The efficacy of project-based learning in enhancing computational thinking among students: A meta-analysis of 31 experiments and quasi-experiments. In *Education and Information Technologies* (Vol. 29, Issue 11). Springer US.

