Al-Adzka: Jurnal Ilmiah Pendidikan Guru Madrasah Ibtidaiyah
Vol. 15, No. 1, 2025
Available online at https://jurnal.uinantasari.ac.id/index.php/adzka
Email: aladzka@uin-antasari.ac.id
p-ISSN: 2088-9801 | e-ISSN: 2597-937X

Coding for Young Learners: Enhancing Computational Thinking and Creativity in Elementary Education

Hilmiyah*[1], Abd Muhith[2], Syamsul Bahri[3]
[1,2,3] Islamic Elementary Teacher Education Program. UIN Kiai Haji Achmad Siddiq Jember, Indonesia

*Correspondence Author: hilmiyahhilmiyah78@gmail.com

Received: 2025-03-23; **Revised:** 2025-04-16; **Accepted**: 2025-05-02; **Published**: 2025-05-05
How to cite: Hilmiyah, Muhith, A., & Bahri, S. (2025). Coding for Young Learners: Enhancing Computational Thinking and Creativity in Elementary Education. *Al-Adzka: Jurnal Ilmiah Pendidikan Guru Madrasah Ibtidaiyah, 15*(1), 96–114. https://doi.org/10.18592/aladzkapgmi.v15i1.16103

Abstract: *Coding skills have become essential for young learners to develop computational thinking and creativity. This study explores the implementation of coding education to enhance computational thinking among elementary school students, particularly at SD Negeri 21 Dauh Puri Denpasar, where technological access is limited. A case study approach was applied with 28 fifth-grade students. Data were collected through observations, interviews, documentation, and questionnaires. Findings revealed increased computational thinking scores from 56.3 to 74.8 after Scratch-based coding lessons. Significant improvements were observed in decomposition, pattern recognition, and algorithmic thinking, although abstraction remained challenging. Students demonstrated high enthusiasm, especially in creating animations and games. However, limited infrastructure and insufficient teacher training hindered optimal implementation. The study concludes that coding education can foster students' critical thinking and creativity but must be supported by structured teacher training and adequate technological resources. It recommends developing innovative and sustainable learning strategies to make coding an integral part of the elementary education curriculum in Indonesia.*
Keywords: *Coding Education, Computational Thinking, Elementary Education, Teacher Training, Digital Learning*

Abstrak: *Keterampilan coding menjadi kompetensi penting bagi siswa sejak usia dini untuk membangun pemikiran komputasional dan kreativitas. Penelitian ini bertujuan mengeksplorasi implementasi pembelajaran coding dalam meningkatkan kemampuan berpikir komputasional siswa sekolah dasar, khususnya di SD Negeri 21 Dauh Puri Denpasar yang memiliki keterbatasan teknologi. Penelitian menggunakan pendekatan studi kasus dengan subjek 28 siswa kelas V. Teknik pengumpulan data meliputi observasi, wawancara, dokumentasi, dan kuesioner. Hasil penelitian menunjukkan peningkatan skor berpikir komputasional dari 56,3 menjadi 74,8 setelah intervensi pembelajaran coding berbasis Scratch. Peningkatan signifikan terjadi pada aspek dekomposisi, pengenalan pola, dan berpikir algoritmik, meskipun aspek abstraksi masih menjadi tantangan. Siswa menunjukkan antusiasme tinggi, terutama dalam membuat animasi dan game. Namun, keterbatasan infrastruktur dan kurangnya pelatihan guru menjadi hambatan utama. Kesimpulannya, pembelajaran coding dapat meningkatkan kemampuan berpikir kritis dan kreativitas siswa, tetapi perlu didukung dengan pelatihan guru yang terstruktur dan penyediaan sarana teknologi yang*

memadai. Penelitian ini merekomendasikan pengembangan strategi pembelajaran inovatif dan berkelanjutan guna menjadikan coding sebagai bagian integral dari kurikulum pendidikan dasar di Indonesia.
Kata Kunci: *Pendidikan Pengkodean, Pemikiran Komputasi, Pendidikan Dasar, Pelatihan Guru, Pembelajaran Digital*

Introduction

In today's digital era, coding skills have become an essential competency for children from an early age, and stated that the concept of computational thinking (CT) encompasses logical reasoning, systematic problem-solving, and the use of algorithms in daily life. Computational thinking is crucial for aspiring programmers and beneficial across various disciplines as it fosters critical and structured thinking (Yildirim & Uluyol, 2023). Furthermore Ramadhan et al., (2023), in constructionism, asserted that children learn more effectively when actively constructing something. In the context of coding education, children are not merely technology users but also creators. Early exposure to coding fosters creativity and problem-solving abilities and enhances logical and systematic thinking skills (Karakaya Cirit, 2022).

Globally, many countries have integrated coding education into their elementary school curricula. For instance, the United Kingdom introduced coding into its curriculum in 2014 for pupils aged five and above (Yang et al., 2023). Similarly, countries such as Finland and Estonia have adopted coding curricula to enhance children's technological skills from an early age (Ergin & Ercan, 2022). Numerous studies highlight that early exposure to coding enhances computational thinking skills and prepares students for the digital age. According to a report by *Code.org* (2021) in (Adigüzel et al., 2023), over 40% of countries worldwide have implemented coding curricula in elementary education. Additionally, research by Totan and Korucu, (2023) revealed that students introduced to coding at an early stage exhibit superior problem-solving abilities, critical thinking, and collaborative skills.

In Indonesia, increasing attention is being given to coding education, particularly with the advent of digitalization. Through the Ministry of Education, Culture, Research, and Technology, the government has developed various policies to support technology-based learning, such as the *Merdeka Belajar* program, which promotes technology integration in education (Tanya et al., 2024). However, the implementation of coding education in Indonesia still faces challenges. Some schools have begun introducing coding, particularly through extracurricular programs or collaborations with private organizations, such as Code for Kids or Google CS First. Nevertheless, access to technological resources and trained educators remains uneven. A study by Putro & Astuti, (2024), indicated that despite growing awareness of the importance of coding, many elementary schools still lack a structured curriculum.

At SD Negeri 21 Dauh Puri Denpasar, technology-based learning encounters several challenges. Preliminary observations indicate limitations in technological resources, such as computers and stable internet access. Additionally, teachers require further training to integrate coding into the curriculum effectively. SD Negeri 21 Dauh Puri Denpasar learners have had minimal exposure to coding concepts. However, the effective implementation of coding education can enhance their computational thinking skills and facilitate the development of interactive learning media. Therefore, a structured coding introduction program is essential to improve learners' and teachers' understanding of this concept.

Several studies have explored the significance of coding education for young learners. Research by Muklason et al., (2023), demonstrated that visual-based coding

programs, such as *Scratch*, enable children to develop computational thinking skills without requiring proficiency in complex programming syntax. Furthermore, research by Samadi & Handayani, (2022) indicated that coding education enhances students' learning motivation and collaborative skills. Learning to code may support students' ability to use logic and reasoning in writing (Thompson & Childers, 2021). Study by Istiqomah & Fanny Novika, (2024) revealed that incorporating coding in project-based learning fosters creativity and enhances students' cognitive skills. However, this study was primarily conducted in schools with sufficient access to technology.

Based on previous studies, coding education significantly benefits young learners, particularly in improving computational thinking and creativity. However, several research gaps require further investigation. Firstly, there is a lack of research in schools with limited technological resources. Most prior studies were conducted in schools with adequate technological facilities, whereas many Indonesian schools, including SD Negeri 21 Dauh Puri Denpasar, still experience resource constraints. Secondly, there is a scarcity of studies on the impact of coding in developing interactive learning media.

Most research focuses on enhancing computational thinking but does not explore how coding can assist students in creating engaging learning materials. Thirdly, there is limited research on teacher training for coding instruction. Although the benefits of coding education are well-documented, few studies examine how teacher training influences the effectiveness of coding education in elementary schools. Therefore, this study aims to address these gaps by exploring how coding education can aid students in developing learning media and enhancing their computational thinking skills, particularly in SD Negeri 21 Dauh Puri Denpasar, where access to technology is limited.

Method

This study employs a case study approach to explore the introduction of coding education for developing learning media and enhancing computational thinking skills among elementary school students. A case study methodology is chosen as it allows for an in-depth examination of the phenomenon within a real-life context (Sugiyono, 2013). The research was conducted at SD Negeri 21 Dauh Puri Denpasar, focusing on fifth-grade learners as the elementary subjects of the study, which had 28 students. The research subjects are fifth-grade pupils at SD Negeri 21 Dauh Puri Denpasar. This grade level was selected based on students' cognitive and developmental readiness to engage in coding activities. Additionally, teachers and school administrators were involved as key informants to provide insights into the implementation of coding education at the school.

Multiple data collection techniques were utilized to obtain comprehensive data, including observation, interviews, documentation, and questionnaires. Classroom observations were conducted to examine pupils engagement and interaction during coding activities, focusing on their problem-solving approaches, collaboration, and computational thinking development. Semi-structured interviews were conducted with teachers and school administrators to gather information on the challenges and opportunities in integrating coding into the curriculum. In contrast, interviews with students help understand their perceptions and experiences in learning coding. Additionally, school records, lesson plans, and student coding projects were analyzed to assess the implementation and outcomes of coding education. In contrast, relevant school policies and previous learning materials were reviewed to provide contextual background. Questionnaires were distributed to students to measure their

computational thinking skills before and after the coding intervention, and additional questionnaires were given to teachers to assess their perceptions of coding education and its impact on student learning.

Table 1. Questionnaire Grid

No	CT Aspect	Indicator	Item Number	Sample Statement
1	Decomposition	Able to break down problems into smaller parts	1	I can break a big problem into smaller steps to solve it.
2		Solve complex tasks step by step	2	I find it easier to solve problems by working through them step by step.
3	Pattern Recognition	Identify patterns in problems	3	I can recognize the same patterns in several tasks or problems.
4		Use known patterns to solve new problems	4	I use the same method when facing similar problems.
5	Algorithmic Thinking	Design systematic steps to solve problems	5	I can create a sequence of steps to solve a problem.
6		Follow algorithms to reach accurate solutions	6	I follow certain steps to get the expected results.
7	Abstraction	Ignore irrelevant information when solving problems	7	I can focus on the important information when solving a problem.
8		Apply one solution to various similar problems	8	I can use the same solution to solve similar problems.

Data collected through various methods are analyzed using qualitative and quantitative techniques. Thematic analysis was used to identify patterns and themes emerging from observational data, interview transcripts, and documentation. In contrast, data coding was conducted to categorize information related to students' engagement, computational thinking development, and challenges in coding education. Descriptive statistical analysis was applied to the questionnaire responses to measure changes in students' computational thinking skills, and comparative analysis was performed to evaluate pre- and post-intervention questionnaire results, determining the impact of coding education. The data analysis technique used to assess and determine the improvement in mastery of the material was conducted through normalized gain analysis (<g>). The normalized gain, or N-gain score, aims to measure the effectiveness of a particular method or treatment used in the study. The N-gain score test calculates the difference between the pre-test and post-test scores. By calculating this difference or gain score, we can determine whether applying a specific method or medium can be considered adequate.
The steps taken to analyze the normalized gain are as follows:

$$N\ Gain \frac{PostestScore - PretestScore}{Ideal\ Score - Pretest\ Score}$$

Determining the average value of the normalized gain score Determining the gain improvement criteria in the following table:

Table 2. Interpretation of Normalized Gain Score

Normalized Gain Score	Criteria
g ≤ 0,3	Low
0,3 < g ≤ 0.70	Medium
0,70 , g ≤ 1,00	High

By employing this methodological approach, the study aimed to provide a comprehensive understanding of how coding education can enhance students' computational thinking and facilitate the development of interactive learning media, particularly in elementary schools with limited access to technology.

Result and Discussion

Result

The findings of this study are based on data obtained through observations, interviews, and questionnaires administered to students and teachers at SD Negeri 21 Dauh Puri Denpasar. The results provide insights into student engagement in coding education, the challenges faced, teacher and student perspectives, supporting and inhibiting factors, and changes in students' computational thinking skills. For the implementation, see Figure 1 below:

Figure 1. Introducing Coding Learning

Observations conducted during the coding lessons indicate student engagement varied throughout the learning process. Initially, 70% of students showed enthusiasm when introduced to coding activities using block-based programming tools such as Scratch. Picture of Scratch below:

Figure 2. Scratch for Coding

Many students actively participated in discussions and eagerly experimented with coding commands. However, as the lessons progressed, 30% of students struggled to

understand logic sequencing and debugging errors, leading to frustration. Additionally, the lack of prior exposure to coding resulted in hesitation among some students, particularly those unfamiliar with digital devices. Technical constraints, such as limited computer access and unstable internet connectivity, also posed challenges, as only one computer was available for every three students, requiring them to take turns. For more details, see the table below:

Table 3. Summary of Readiness to Participate in Coding Activities

Readiness Category	Number of Students	Percentage (%)
Very Unprepared	0	0%
Unprepared	3	10.70%
Fairly Prepared	9	32.10%
Prepared	13	46.40%
Very Prepared	3	10.70%
Total	28	100%

The readiness survey results showed that most students demonstrated a positive attitude toward participating in coding activities. Of 28 students, 46.4% were categorized as Prepared, 32.1% were Fairly Prepared, and 10.7% were Very Prepared. However, 10.7% of students were still categorized as Unprepared, and none were classified as Very Unprepared. These findings indicate that although most students were open and ready to engage with coding education, a small portion still required additional support and encouragement to build their confidence and basic skills before fully participating in coding activities.

Meanwhile, evaluating students' computational thinking (CT) abilities revealed varied strengths across the four measured aspects: decomposition, pattern recognition, algorithmic thinking, and abstraction. In the Decomposition and Pattern Recognition aspects, 46.4% of students were categorized as Strong, with an additional 32.1% being Fairly Strong, and only 10.7% considered Weak. A similar trend was observed in Algorithmic Thinking, where 53.6% of students reached the Strong category, the highest among all aspects, reflecting their improved ability to design systematic steps in problem-solving after coding lessons. However, the Abstraction aspect showed a different pattern, with 46.4% of students rated as Fairly Strong, 21.4% as Weak, and 10.7% still in the Very Weak category. Suggests that abstraction, which requires students to generalize solutions and ignore irrelevant details, remains the most challenging skill to develop among young learners. For more detail in the table below:

Table 4. Summary of Computational Thinking Strengths

Aspect	Very Weak	Weak	Fairly Strong	Strong	Very Strong
Decomposition	0 (0%)	3 (10.7%)	9 (32.1%)	13 (46.4%)	3 (10.7%)
Pattern Recognition	0 (0%)	3 (10.7%)	9 (32.1%)	13 (46.4%)	3 (10.7%)
Algorithmic Thinking	0 (0%)	3 (10.7%)	7 (25.0%)	15 (53.6%)	3 (10.7%)
Abstraction	3 (10.7%)	6 (21.4%)	13 (46.4%)	6 (21.4%)	0 (0%)

Interviews with teachers revealed mixed reactions toward coding education. Teachers acknowledged the potential benefits of coding in enhancing students' problem-solving and logical reasoning skills. However, they also expressed concerns regarding their preparedness in teaching coding effectively. One teacher stated,

> "We see coding as an important skill for the future, but many of us lack the necessary training to teach it effectively." (W.T1/15-01-2025)

Meanwhile, students shared varied responses. Some expressed excitement about creating interactive animations and games, while others found certain aspects of coding, such as debugging, challenging. A fifth-grade student commented,

> "I like making characters move, but sometimes, when the program does not work, I do not know how to fix it." (W.T1/15-01-2025)

The study also identified several supporting and inhibiting factors in implementing coding education. The elementary supporting factor was the students' intrinsic motivation and curiosity about technology, which helped sustain their engagement.

> "I enjoy making animations, but debugging is difficult. It's fun when the program works, but frustrating when I get errors." (W.S1&S2/15-01-2025)

Additionally, external resources, such as online coding tutorials and support from technology-savvy teachers, facilitated learning. On the other hand, inhibiting factors included inadequate technological infrastructure, limited teacher training, and time constraints within the existing curriculum. Teachers expressed needing more structured professional development programs to improve their coding instruction skills. One teacher suggested,

> "If we had more workshops and access to teaching materials, we could integrate coding more effectively into our lessons." (W.T2/15-01-2025)

The questionnaire results demonstrated significant improvements in students' computational thinking skills. Before the intervention, the average computational thinking score was 56.3 (on a 100-point scale). After participating in the coding lessons, the average score increased to 74.8, reflecting a 32.8% improvement. Specifically, students showed notable progress in decomposition (breaking down problems into smaller steps), pattern recognition, and algorithmic thinking. However, abstraction and the ability to generalize and remove unnecessary details remained challenging for many students.

Table 5. The questionnaire results

Computational Thinking Aspect	Pre-Test	Post-Test	N-Gain	N-Gain Category
Decomposition	58,2	75,4	0,41	Medium
Pattern Recognition	55,1	72,3	0,38	Medium
Algorithmic Thinking	57,5	76,8	0,45	Medium
Abstraction	54,4	68,2	0,30	Medium

Table 5 illustrates the questionnaire results that assessed students' computational thinking skills before and after implementing the learning intervention. The data highlight significant improvements across all four core aspects of computational thinking: decomposition, pattern recognition, algorithmic thinking, and abstraction.

In the decomposition aspect, the average score increased from 58.2 in the pre-test to 75.4 in the post-test. Indicates that after the intervention, students could break down complex problems into smaller, manageable parts. In pattern recognition, the score rose from 55.1 to 72.3, showing that students improved in identifying similarities,

trends, or repeated structures in data or problems, an essential skill in programming and logical analysis. The most notable improvement was algorithmic thinking, which increased from 57.5 to 76.8. Suggests students develop a better ability to formulate step-by-step solutions or procedures to solve specific tasks.

Meanwhile, the abstraction aspect, which involves focusing on important information and ignoring irrelevant details, also showed progress, with scores rising from 54.4 to 68.2. Although this was the lowest post-test score among the four aspects, the increase demonstrates a positive learning outcome. The average score across all aspects improved significantly from 56.3 to 74.8, indicating that the learning intervention substantially enhanced students' computational thinking abilities. These improvements suggest a better understanding of computational concepts and stronger analytical and problem-solving skills, crucial for success in computer science and related disciplines.

The findings indicate that while coding education enhances students' computational thinking skills and engagement, its implementation requires addressing teacher preparedness and infrastructure support challenges.

Discussion

Enthusiasm in Learning Coding with Scratch in Elementary

Coding education is increasingly recognized as an essential skill in 21st-century education. Computational thinking skills developed through coding can enhance students' problem-solving abilities, logical reasoning, and creativity (Metin et al., 2023). In the context of coding education at SD Negeri 21 Dauh Puri Denpasar, findings indicate that student engagement varies. These findings align with previous studies suggesting that technological experience, teacher preparedness, and infrastructure play crucial roles in the successful implementation of coding in elementary schools (Wooten et al., 2024).

Observations showed that most students (70%) exhibited high enthusiasm when introduced to coding using Scratch. According to the research by Calder and Rhodes, (2023), block-based programming environments can increase student interest and motivation due to their visual and interactive nature. Block-based programming approaches like Scratch provide a more intuitive experience than text-based programming. Students can drag and arrange code blocks directly without worrying about complex syntax in this environment. Such an approach enables students to concentrate on problem-solving and core programming concepts, which are typically challenging in traditional text-based coding.

However, as task complexity increased, student engagement declined, particularly when faced with debugging challenges and difficult logic sequences. Suggests that while Scratch makes grasping the basics of programming easier, problem-solving challenges still significantly affect student engagement levels. According to Vygotsky's theory of cognitive development Yu et al., (2013), effective learning occurs within the Zone of Proximal Development (ZPD). In this context, students can complete tasks with assistance or support from teachers and peers. In other words, some students may struggle to understand more abstract programming concepts without proper guidance.

The data indicates that students with prior technological experience grasp coding concepts more quickly than those without such backgrounds. Prior exposure to technology contributes to a faster understanding of programming concepts. Conversely, students without previous experience require more intensive guidance to build foundational knowledge before tackling more complex tasks. The role of

teachers as facilitators is crucial. Teachers must provide scaffolding, which is gradual assistance tailored to students' understanding levels. This approach may include giving hints, offering simpler code examples, or encouraging group work to help students facing difficulties.

According to Dia et al., (2024), computational thinking develops gradually. This process starts with basic exploration before students fully understand complex abstract concepts. When first introduced to Scratch, students focus more on exploratory aspects, such as experimenting with different code blocks and observing their effects on animations or character movements. Students begin recognizing patterns and logic behind programming, such as loops, conditional statements, and debugging. However, challenges arise when applying these concepts in more complex scenarios. At this stage, students who have already built a programming foundation can transition more easily from mere exploration to systematic problem-solving, whereas beginners may struggle and lose motivation.

Learning strategies that align with students' developmental stages are essential to address these challenges. Teachers can implement project-based learning, where students can create projects based on their interests. This approach helps maintain motivation and engagement, as students feel a sense of ownership over their work. Using Scratch as a programming learning tool has proven effective in increasing student enthusiasm, especially at the introductory stage. However, challenges remain, particularly regarding task complexity, which can lead to declining student engagement. By applying the ZPD and scaffolding approach and understanding the gradual development of computational thinking, teachers can support students in overcoming difficulties and maintaining their motivation to learn to code.

Challenges in Coding Education in Elementary

This study identifies several key challenges in coding education, including a lack of prior exposure to coding, difficulties in debugging, limited infrastructure, and unstable internet connectivity. These factors significantly impact students' ability to learn and engage with coding concepts effectively. Understanding these challenges in detail can help educators and policymakers develop strategies to improve coding education.

1. Lack of Prior Exposure to Coding

One of the most fundamental challenges is that many students have never been exposed to coding. Without prior experience, students often struggle to grasp essential programming concepts, such as sequencing, loops, and conditionals. Unlike subjects such as mathematics or science, which students typically encounter from an early age, coding is a relatively new addition to many curricula.

Students without prior experience may feel intimidated by programming tasks, especially when encountering unfamiliar terminologies and abstract logic. This challenge aligns with Vygotsky's Zone of Proximal Development (ZPD) Smith, (1993), which suggests that learners require scaffolding and support from teachers or peers to bridge the gap between what they already know and what they need to learn. Students struggle to progress beyond basic coding exercises without proper guidance and foundational knowledge.

Additionally, students with prior technological experience, such as familiarity with educational games or block-based platforms like Scratch, tend to adapt more quickly to coding concepts. This disparity creates a learning gap, where some students advance faster while others struggle to keep up. Educators must integrate

introductory activities focusing on algorithmic thinking and problem-solving to address this issue before diving into actual coding exercises.

2. Difficulties in Debugging

Debugging is another elementary challenge students face, requiring analytical thinking and attention to detail. Muhammad et al., 2023), Highlight that debugging is one of the most challenging aspects of computational thinking for novice students. Students who write code often struggle to identify and fix errors, leading to frustration and disengagement. Several factors contribute to debugging difficulties in the concept map below:

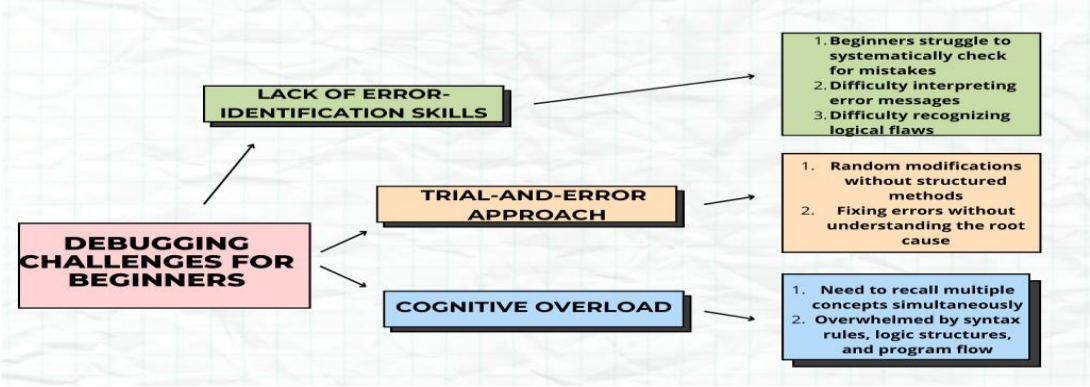

Figure 4. Debugging Challenges for Beginners

The concept map illustrates three elementary challenges beginners face when engaging in debugging activities. First, a lack of error-identification skills leads to difficulties in systematically checking code. Many students cannot interpret error messages or recognize where logical flaws occur in their programs, making the debugging process frustrating and inefficient. Second, students who lack a structured debugging strategy often employ the trial-and-error approach. Instead of methodically analyzing the code, they randomly alter parts of it in hopes of solving the problem without understanding the underlying issue. This approach often leads to more confusion and does not build lasting problem-solving skills. Third, cognitive overload heavily impacts beginners because debugging requires them to recall and integrate various programming elements simultaneously. Students must manage syntax accuracy, logical coherence, and the sequence of operations, which can be mentally exhausting for those still mastering basic coding concepts. To improve debugging skills, educators can introduce debugging strategies, such as the concept map below

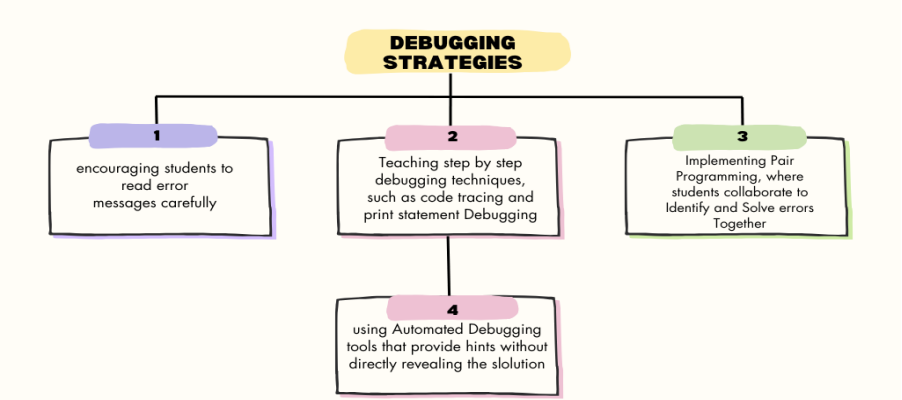

Figure 5. Debugging Strategies

The concept map illustrates four key methods educators can implement to enhance students' debugging skills. First, encouraging students to read error messages carefully helps them develop a more systematic approach to identifying problems in their code. Students can more accurately pinpoint the source of issues by understanding and analyzing error messages instead of ignoring or guessing their meaning. Second, teaching step-by-step debugging techniques, such as code tracing (following code execution line by line) and print statement debugging (inserting print commands to monitor program behavior), provides students with structured methods to investigate and solve problems. The process helps develop logical thinking skills and enhances comprehension of program flow.

Third, implementing pair programming encourages collaboration, allowing students to work together to identify and fix coding errors. Students can learn from one another through this cooperative method, discuss solutions, and develop stronger problem-solving strategies. Finally, using automated debugging tools that provide hints without revealing solutions can guide students toward finding errors independently. These tools maintain the debugging challenge while offering subtle support, which helps students build resilience and autonomy in problem-solving.

By incorporating these strategies, students can develop problem-solving skills and gain confidence in troubleshooting their code.

Another critical challenge is inadequate infrastructure, including insufficient computers, outdated hardware, and unstable internet connectivity. Constructionism theory emphasizes the importance of technology access in fostering hands-on learning and exploration (Kuleli, 2024). However, without sufficient resources, students struggle to gain meaningful coding experience.

According to Gümüş et al., (2024), resource constraints are a significant barrier to effective coding education in elementary schools. Educators can implement effective and structured strategies to enhance students' debugging skills. One important approach is encouraging students to read error messages carefully. By developing the habit of thoroughly understanding these messages, students can more easily identify the root causes of problems in their code. In addition, teachers can introduce step-by-step debugging techniques, such as code tracing and print statement debugging, which help students follow the program's flow and systematically locate errors. Another valuable method is implementing pair programming, where students work together to identify and solve bugs. This collaborative approach speeds debugging and strengthens students' communication skills and critical thinking. Automated debugging tools that offer hints without directly revealing the solution can further empower students to think independently and deepen their problem-solving abilities. Using these strategies, students can develop stronger, more confident debugging skills.

The lack of resources significantly affects students' ability to develop coding skills. Without consistent practice, they may struggle to retain concepts and build fluency in programming. Additionally, a lack of personal devices reduces opportunities for students to explore coding outside of class, limiting their ability to experiment and develop creative projects. Schools and policymakers are encouraged to explore strategic solutions to address these challenges.

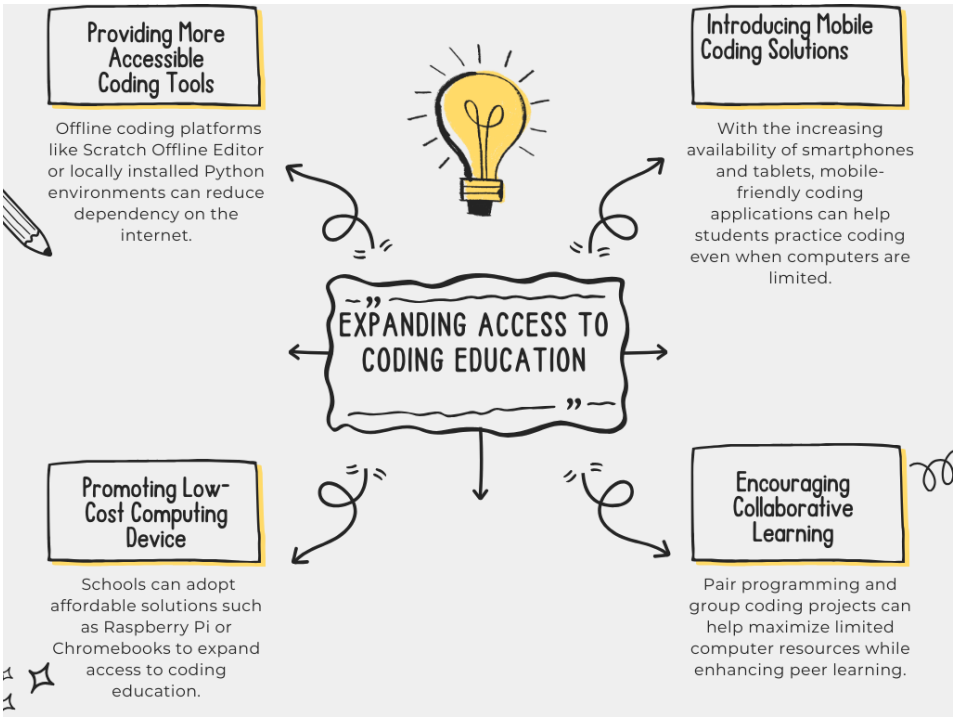

Figure 6. Expanding Acces to Coding Education

In summary, coding education faces several challenges, including a lack of prior exposure, debugging difficulties, and infrastructure constraints. Debugging remains a significant obstacle for novice programmers, who often struggle to identify and fix errors. Moreover, inadequate infrastructure, such as limited computer access and unstable internet, further restricts students' ability to practice and explore coding effectively.

To address these issues, educators and institutions should focus on building foundational coding knowledge, integrating structured debugging strategies, and improving access to technological resources. By implementing these solutions, coding education can become more inclusive, engaging, and effective for all students, regardless of their prior experience or learning environment.

Teacher and Student Perspectives Coding Education in Elementary

1. Teacher Perspective

Although teachers recognize the importance of coding in enhancing students' logical thinking and problem-solving skills, many feel unprepared to teach it effectively. This challenge is particularly evident among teachers without a computer science background, making it difficult for them to explain coding concepts and provide adequate guidance to students systematically. As a result, coding education in schools often remains superficial, with students only being introduced to basic concepts without a deeper understanding of how to apply them in real-world contexts. According to the findings Iyamuremye & Nsabayezu, (2022), the lack of teacher training in coding is one of the elementary obstacles to the successful implementation of computational education at the elementary level. Teachers may struggle to develop lesson plans that effectively integrate coding into their curriculum without proper training, leading to inconsistent and less effective instruction.

Furthermore, many teachers have expressed that additional training and professional development opportunities, such as workshops and hands-on practice sessions, would significantly boost their confidence and ability to teach coding. This perspective is supported by Özkan et al., (2022), who found that professional development programs are crucial in improving teachers' effectiveness in coding instruction. Through structured training, teachers can learn coding concepts more thoroughly and acquire strategies to integrate computational thinking into various subjects, such as mathematics, science, and language arts. This interdisciplinary approach allows students to see coding as a valuable tool beyond computer science, fostering a more holistic educational experience.

Moreover, investing in teacher training programs can have long-term benefits for both educators and students. Teachers who feel competent in teaching coding are likelier to implement engaging and innovative teaching methods, such as project-based learning, gamification, and real-world coding applications. These methods can enhance student engagement and motivation, improving learning outcomes. Additionally, well-trained teachers can help bridge the digital divide by ensuring all students have access to quality coding education regardless of their background. Given the increasing relevance of coding skills in the modern workforce, schools must prioritize teacher training to equip students with the necessary skills for future careers in technology and beyond.

While teachers acknowledge the value of coding education, their lack of preparation and training remains a significant barrier to effective instruction. Addressing this issue requires comprehensive professional development programs that equip teachers with technical knowledge and pedagogical strategies. By providing adequate training and support, schools can empower educators to deliver high-quality coding education, benefiting students by enhancing their computational thinking abilities and preparing them for a technology-driven world.

2. Student Perspectives

Students exhibited a wide range of responses to coding education, reflecting both enthusiasm and frustration. Many students found coding exciting and engaging, particularly because of its creative aspects, such as designing animations, developing interactive stories, and programming simple games. These creative opportunities allowed students to express their ideas uniquely and gave them a sense of accomplishment when successfully implementing their projects. However, despite their initial excitement, many students also experienced frustration, particularly when encountering program errors or debugging their code. This frustration often stemmed from difficulties identifying mistakes, understanding error messages, and applying logical thinking to troubleshoot problems effectively.

Jayanti et al., (2021) Emphasized that students' early experiences with coding play a crucial role in shaping their long-term attitudes toward programming. If students find coding too difficult or encounter repeated failures without sufficient support, they may develop negative perceptions of the subject, leading to a decline in motivation and engagement. On the other hand, when students receive appropriate guidance and encouragement, they are more likely to persevere through challenges and develop a positive attitude toward coding. A supportive learning environment where students can experiment, make mistakes, and steadily build their problem-solving confidence is essential for effective learning.

Another significant factor influencing students' experiences with coding is their prior exposure to technology. Students with little or no experience with computers or digital tools often struggled to grasp fundamental coding concepts. These students

struggled with basic programming logic, such as sequencing commands, using loops and conditionals, and understanding how different program components interact. Research by Putro and Astuti, (2024) supports this observation, showing that early exposure to technology plays a key role in students' confidence and success in learning to code. Students familiar with technology from an early age tend to develop stronger digital literacy skills, making it easier to adapt to coding and computational thinking.

Moreover, students from technologically disadvantaged backgrounds may require additional support to bridge the gap in their learning. Without prior exposure to technology, these students might feel intimidated by coding and perceive it as an overly complex subject. Educators must adopt differentiated instruction strategies catering to varying technological familiarity levels to address this issue. For example, incorporating visual programming languages like Scratch or Blockly can help beginners develop a foundational understanding of coding concepts before transitioning to text-based programming languages like Python or JavaScript. Additionally, providing step-by-step tutorials, interactive coding exercises, and peer mentoring programs can help struggling students build their confidence and skills at a comfortable pace.

In addition to technical challenges, the learning environment also plays a crucial role in shaping students' attitudes toward coding. A collaborative, hands-on approach to coding education can significantly enhance student engagement and motivation. When students work on projects, they can share ideas, troubleshoot problems collectively, and learn from each other's experiences. Pair programming and group coding activities encourage teamwork and communication, making learning more interactive and less intimidating. Teachers can also incorporate real-world coding applications, such as programming simple robots, creating mobile apps, or designing websites, to demonstrate the practical relevance of coding skills. By connecting coding education to real-world scenarios, students can see the value of their learning and remain motivated to develop their skills further.

Gamification can be an effective strategy to sustain students' interest in coding. Integrating game-like elements such as coding challenges, badges, leaderboards, and rewards can make learning more enjoyable and encourage students to persist through complex tasks. Platforms like Code.org, Scratch, and Tynker utilize gamification techniques to make coding more accessible and engaging for students of all skill levels. When students view coding as fun and rewarding, they are more likely to stay motivated and continue exploring computational thinking beyond the classroom.

Students' experiences with coding education vary widely, with enjoyment often stemming from creative opportunities and frustration arising from technical challenges. Prior exposure to technology plays a significant role in determining students' confidence and success in learning to code, with those lacking technological experience requiring additional support. Educators must create an inclusive and engaging learning environment that includes structured guidance, collaborative learning opportunities, and interactive tools to ensure all students develop a positive attitude toward coding. By fostering a supportive coding education framework, schools can equip students with essential computational thinking skills, preparing them for future academic and career opportunities in the digital era.

Changes in Computational Thinking Skills of elementary students

The questionnaire results reveal significant improvements in students' computational thinking skills, reinforcing existing research on the positive impact of coding

education on cognitive development. According to the findings of Muklason et al., (2023), who identified four key areas of computational thinking that coding helps develop: problem decomposition, pattern recognition, algorithmic thinking, and abstraction. Students demonstrated notable progress in breaking down complex problems into smaller, more manageable components, allowing them to approach coding tasks systematically rather than feeling overwhelmed by their complexity. Additionally, they improved their ability to identify recurring patterns in code and apply previously learned solutions to new problems, an essential skill in both programming and broader problem-solving contexts.

Furthermore, students showed enhanced algorithmic thinking as they became more adept at structuring their code logically, using loops, conditionals, and functions to create efficient and reusable solutions. This skill not only aids in coding but also strengthens logical reasoning and step-by-step planning, which are crucial for mathematics, science, and other analytical disciplines. However, despite these improvements, abstraction remained a significant challenge for many students. This difficulty aligns with the observations of Theelen et al., (2024), who found that abstraction, the ability to generalize solutions and apply them across different contexts, is often one of the most complex aspects of computational thinking for beginners. Many students struggled with understanding how specific pieces of code could be modified or reused in different scenarios, often relying on direct replication rather than conceptual generalization.

The increased computational thinking scores from 56.3 to 74.8 after Scratch-based coding lessons. This substantial improvement suggests that, despite the inherent challenges in learning to code, students gained meaningful benefits from their experiences. According to the research of Zeng and Zhu (2021), early exposure to coding strengthens computational thinking skills and enhances students' preparedness for future academic and career challenges. The growth in students' scores reflects their ability to grasp coding concepts and their increased confidence in applying computational thinking strategies to solve problems.

One of the key factors contributing to these improvements is the hands-on, interactive nature of coding education. Unlike traditional learning methods that rely heavily on passive instruction, coding encourages active engagement through experimentation, debugging, and iterative problem-solving. Students learn through trial and error, testing their ideas and refining their code based on feedback. This iterative process helps reinforce computational thinking skills and fosters resilience in problem-solving, as students become accustomed to encountering errors and working through them rather than becoming discouraged.

Additionally, structured guidance and instructional support cannot be overlooked when explaining the observed improvements. Many students benefited from scaffolded learning approaches, where foundational concepts were introduced gradually before progressing to more complex topics. Visual programming environments like Scratch likely played a crucial role in making abstract concepts more tangible, allowing students to manipulate coding elements through drag-and-drop interfaces before transitioning to text-based programming. This step-by-step approach helped students build confidence and understand computational thinking principles more deeply.

Moreover, collaborative learning contributed to students' progress. Group coding activities, peer programming, and classroom discussions allowed students to learn from one another, share problem-solving strategies, and refine their understanding through social interaction (Fitriyah, 2024). Such an approach aligns with the

constructivist view of learning, which suggests that knowledge is best acquired through active participation and collaboration rather than passive reception. When students explain their reasoning to peers or work together on projects, they reinforce their understanding while gaining new perspectives on approaching coding challenges (Kirom & Rosyida, 2024).

Despite these positive outcomes, students' difficulty with abstraction suggests that additional instructional strategies may be necessary to enhance this skill further. Teachers could incorporate more explicit examples of abstraction in coding lessons, such as demonstrating how functions and variables can generalize solutions across multiple scenarios (Khairullina & Prastowo, 2024). Additionally, integrating computational thinking exercises that emphasize pattern recognition and generalizations, such as creating reusable code modules or modifying existing algorithms for new applications, could help students develop a more flexible approach to problem-solving.

The long-term implications of these findings are significant. As computational thinking becomes an increasingly valuable skill in a technology-driven world, early exposure to coding can provide students with a strong foundation for future academic and career opportunities. Coding education prepares students for computer science-related fields and disciplines that require logical reasoning and analytical thinking, such as engineering, data science, and finance, by equipping students with problem decomposition, pattern recognition, algorithmic thinking, and abstraction skills.

The questionnaire results underscore the effectiveness of coding education in developing students' computational thinking abilities, with notable improvements in problem decomposition, pattern recognition, and algorithmic thinking. While abstraction remains a challenge, the overall increase in scores from 56.3 to 74.8 indicates that students benefited significantly from their coding experience. These findings highlight the importance of early coding education, structured instructional support, and collaborative learning environments in fostering computational thinking skills. Refining teaching strategies to address abstraction challenges further enhances students' ability to generalize and apply coding concepts across different contexts, ensuring they are well-prepared for the demands of an increasingly digital world.

Coding education at SD Negeri 21 Dauh Puri Denpasar has positively impacted students' computational thinking skills. However, challenges such as teacher preparedness and infrastructure must be addressed to enhance the effectiveness of coding implementation. With the proper support, coding can become an integral part of elementary education and help students develop the 21st-century skills they need for the future.

Conclusion

This study shows that coding education positively impacts the development of computational thinking and creativity in elementary school. The increased computational thinking scores from 56.3 to 74.8 after Scratch-based coding lessons. Student enthusiasm was high, especially in creative activities such as creating animations and games. However, limited technology infrastructure and teacher readiness in teaching coding are still barriers to optimal implementation. Teachers recognize the importance of coding in equipping students with 21st-century skills but feel underprepared due to the lack of formal training. Therefore, more structured training programs and adequate resource support are needed to ensure that coding education can be effectively integrated into the elementary school curriculum. In

addition, there is a need for innovative strategies to overcome technological limitations, such as the use of more flexible devices or project-based learning that does not always rely on computers. By rectifying these challenges, coding education in elementary schools can be an effective means of building students' critical thinking, problem-solving, and creativity skills. Further research is needed to explore more inclusive coding learning methods and measure the long-term impact of coding education on students' academic and non-academic development. The results suggest the need for further investment in teacher training and digital resources to maximize the impact of coding education in elementary schools with limited technological access.

References

[1] Adigüzel, S., Eryilmaz, S., Gencer, T., & Göksu, H. (2023). Coding activities in IT courses through the lenses of IT teachers. *Journal of Educational Technology and Online Learning*, *6*(2), 384–402. https://doi.org/10.31681/jetol.1219963

[2] Calder, N., & Rhodes, K. (2023). Coding and mathematics: How did coding and collaboration facilitate thinking? *Australasian Journal of Technology Education*, *8*, 98–105. https://doi.org/10.15663/ajte.v8i.90

[3] Dia, I. O., Putra, Z. H., Witri, G., Dahnilsyah, & Aljarrah, A. (2024). Development of a Traditional Game-Based Computational Thinking Supplementary Textbook for Elementary School Students. *Mathematics Teaching-Research Journal*, *16*(2), 185–206. https://files.eric.ed.gov/fulltext/EJ1442388.pdf

[4] Ergin, A. Z., & Ercan, Z. G. (2022). The Coding Skills of Pre-School Teacher Candidates. *International Journal of Curriculum and Instruction*, *14*(1), 1052–1071. https://ijci.net/index.php/IJCI/article/view/801

[5] Fitriyah, R. (2024). Effectiveness of Canva-Based Interactive Animation Media in Enhancing Learning Interest of Second-Grade Madrasah Ibtidaiyah Students. *Al-Adzka : Jurnal Ilmiah Pendidikan Guru Madrasah Ibtidaiyah*. *14*(2), 135–150. https://doi.org/10.18952/aladzkapgmi.v12i2.13896

[6] Gümüş, M. M., Kukul, V., & Korkmaz, Ö. (2024). Relationships between middle school students' digital literacy skills, computer programming self-efficacy, and computational thinking self-efficacy. *Informatics in Education*, *23*(3), 571–592. https://doi.org/10.15388/infedu.2024.20

[7] Istiqomah, N., & Fanny Novika. (2024). Pengenalan Coding Membuat Game pada Siswa Sekolah Dasar menggunakan Scratch. *JURPIKAT (Jurnal Pengabdian Kepada Masyarakat)*, *5*(3), 925–938. https://doi.org/10.37339/jurpikat.v5i3.1827

[8] Iyamuremye, A., & Nsabayezu, E. (2022). Mathematics and Science Teacher's Conception and Reflection on Computer Programming with Scratch: Technological and Pedagogical Standpoint. *International Journal of Education, Training and Learning*, *6*(1), 11–16. https://doi.org/10.33094/ijetl.v6i1.488

[9] Jayanti, D., Septiani, J. I., Sayekti, I. C., Prasojo, I., & Yuliana, I. (2021). Pengenalan Game Edukasi sebagai Digital Learning Culture pada Pembelajaran Sekolah Dasar. *Buletin KKN Pendidikan*, *3*(2), 184–193. https://doi.org/10.23917/bkkndik.v3i2.15735

[10] Karakaya Cirit, D. (2022). Coding in Preschool Science and Mathematics Teaching: Analysis of Scratch Projects of Undergraduate Students. *International Journal of Contemporary Educational Research*, *9*(3), 460–475. https://doi.org/10.33200/ijcer.1031848

[11] Khairullina, A., & Prastowo, A. (2024). Implementation of ICT-Based Problem-Based Learning Model to Improve Higher Order Thinking Skills in IPAS Learning in Madrasah Ibtidaiyah. *Al-Adzka : Jurnal Ilmiah Pendidikan Guru Madrasah Ibtidaiyah*. *14*(2), 151–161.

https://doi.org/10.18952/aladzkapgmi.v14i2.13282
[12] Kirom, S., & Rosyida, D. A. (2024). Educational Game " Sianting " to Improve Literacy and Anti- Bullying Education Using Problem-Solving Approach. *Al-Adzka : Jurnal Ilmiah Pendidikan Guru Madrasah Ibtidaiyah*. *14*(2), 189–205. https://doi.org/10.18952/aladzkapgmi.v14i2.14031
[13] Kuleli, S. (2024). *An Examination of the Self-Efficacy Perceptions of Eighth-Grade Students Regarding Computational Thinking Competencies*. *23*(4), 21–38.
[14] Metin, S., Basaran, M., & Kalyenci, D. (2023). Examining coding skills of five-year-old children. *Pedagogical Research*, *8*(2), em0154. https://doi.org/10.29333/pr/12802
[15] Muhammad, I., Rusyid, H. K., Maharani, S., & Angraini, L. M. (2023). Computational Thinking Research in Mathematics Learning in the Last Decade: A Bibliometric Review. *International Journal of Education in Mathematics, Science and Technology*, *12*(1), 178–202. https://doi.org/10.46328/ijemst.3086
[16] Muklason, A., Riksakomara, E., Mahananto, F., Djunaidy, A., Vinarti, R. A., Anggraeni, W., Nurita, R. T., Utamima, A., Fauzia, R., Theresia, L. W., Fikri, M. A., Propitadewa, H., Habibah, J. H., Prasetyo, J. D., Permatasari, S. T. I., Risnina, N. N., Tsaniyah, N. D., & Maulana, M. D. (2023). Coding for Kids: Pengenalan Pemrograman untuk Anak Sekolah Dasar sebagai Literasi Digital Baru di Industri 4.0. *Sewagati*, *7*(3). https://doi.org/10.12962/j26139960.v7i3.506
[17] Özkan, N., Özgeldi, M., & Uzun, E. (2022). 8th Graders' Interpretation of Equal Sign in Scratch: Pan Balance Activities. *Education Quarterly Reviews*, *5*(4), 79–97. https://doi.org/10.31014/aior.1993.05.04.607
[18] Putro, Y. T. M., & Astuti, R. (2024). Penerapan Scratch dalam Pembelajaran Coding Siswa Sekolah Dasar. *Emergent Journal of Educational Discoveries and Lifelong Learning (EJEDL)*, *1*(4), 21. https://doi.org/10.47134/emergent.v1i4.37
[19] Ramadhan, F. A., Muhith, A., Usriyah, L., Islam, U., Kiai, N., Achmad, H., & Jember, S. (2023). Design of Electronic-Based Handout Teaching Materials with Problem-Based Learning Nuances in Learning Mathematics. *EDUHUMANIORA : Jurnal Pendidikan Dasar,* *15*(2), 121–134. https://doi.org/10.17509/eh.v15i2.56031
[20] Samadi, M., & Handayani, A. N. (2022). Media Pembelajaran Interaktif Coding'Bahasa C'untuk Anak-anak Berbasis Aplikasi Augmented Reality dalam Menyongsong. *Jurnal Inovasi Teknologi Dan Edukasi*, *2*(7), 326–338. https://doi.org/10.17977/um068v1i72022p326-338
[21] Smith, A. B. (1993). Early childhood educare: Seeking a theoretical framework in vygotsky's work. *International Journal of Early Years Education*, *1*(1), 47–62. https://doi.org/10.1080/0966976930010105
[22] Sugiyono. (2013). *Metode Penelitian Pendidikan (Pendekatan Kuantitatif, Kualitatif, dan R&D)*. Alfabeta.
[23] Tanya, M., Diah, K., Rahmah, A., & Florian, H. (2024). *Pelatihan Coding Berbasis Project Based Learning (PjBL) Menggunakan Platform Scratch untuk Sekolah Dasar*. *3*(5), 283–291. https://doi.org/10.55824/jpm.v3i5.437
[24] Theelen, H., Vreuls, J., & Rutten, J. (2024). Doing Research with Help from ChatGPT: Promising Examples for Coding and Inter-Rater Reliability. *International Journal of Technology in Education*, *7*(1), 1–18. https://doi.org/10.46328/ijte.537
[25] Thompson, J., & Childers, G. (2021). The impact of learning to code on elementary students' writing skills. *Computers & Education*, *175*, 104336. https://doi.org/10.1016/j.compedu.2021.104336

[26] Totan, H. N., & Korucu, A. T. (2023). The Effect of Block Based Coding Education on the Students' Attitudes about the Secondary School Students' Computational Learning Skills and Coding Learning: Blocky Sample. *Participatory Educational Research*, *10*(1), 443–461. https://doi.org/10.17275/per.23.24.10.1

[27] Wooten, J. O., Cuevas, J. A., The, J. A., Wooten, J. O., & Cuevas, J. A. (2024). The Effects of Dual Coding Theory on Social Studies Vocabulary and Comprehension in Elementary Education. *International Journal on Social and Education Sciences, 6 (4)*. https://doi.org/10.46328/ijonses.696

[28] Yang, Z., Shaffer, P. M., Hagan, C., Dubash, P., & Bers, M. (2023). *Impact Study of the Coding as Another Language Curriculum. March*. https://eric.ed.gov/?id=ED626865

[29] Yildirim, E., & Uluyol, Ç. (2023). Developing Computational Thinking Scale for Elementary School Students and Examining Students' Thinking Levels According to Different Variables. *Journal of Learning and Teaching in Digital Age*, *8*(1), 113–123. https://doi.org/10.53850/joltida.1176173

[30] Yu, Y. H., Hu, Y. N., & Zhang, J. S. (2013). Vygotsky's Zone of Proximal Development: Instructional Implications and Teachers' Professional Development. *Applied Mechanics and Materials*, *411–414*(4), 2952–2956. https://doi.org/10.4028/www.scientific.net/AMM.411-414.2952

[31] Zeng, M., & Zhu, F. (2021). Secure Coding in Five Steps. *Journal of Cybersecurity Education, Research and Practice*, *2021*(1). https://doi.org/10.62915/2472-2707.1076