DEVELOPMENT A CODE VISUALIZATION TOOL ON LOOPING MATERIAL SPECIFICALLY "*FOR*" AND "*WHILE*" TO IMPROVE STUDENTS' LEARNING OUTCOMES.

Manuscript Info

Manuscript History
Received: xxxxxxxxxxxxxxx
Final Accepted: xxxxxxxxxxxx
Published: xxxxxxxxxxxxxxx

Key words:-
Code visualization tool,
Programming,
Algorithm,
For,While

Abstract

This research aims to develop a web-based Code Visualization Tool specifically designed to help students understand the concept of looping in programming, particularly the "*for"* and "*while"* statements. The study employed a Research and Development (R&D) approach using the ADDIE model, consisting of five stages: Analysis, Design, Development, Implementation, and Evaluation. Student responses are positive, showing usability and visual appeal levels above 80%. The effectiveness test revealed that the average pre-test score increased from 42.5 to 70 in the post-test, resulting in an N-Gain score of 0.48 (medium category). These results demonstrate that the Code Visualization Tool has a moderate effectiveness in improving students' learning outcome of looping concepts.

Introduction: -

The Algorithm and Programming course is one of the fundamental subjects that plays a crucial role in the information technology curriculum (Fitri et al., 2025; H. C. B. Chan et al., 2024; Manorat et al., 2025; Z. Wu & Wan, 2025; Yuricha& Phan, 2023). At the initial stage, students are introduced to the concept of looping or iteration, which serves as a foundation in computer programming. (Izul et al., 2024). However, looping material is often considered difficult by 66.7% of students out of 15 Information Technology Education students interviewed, as it involves abstract logic, repetitive instructions, and dynamic variable interactions during the program execution process. This causes some students to merely memorize the syntax without truly understanding how the looping flow works within the code (Permatasari et al., 2018; Syafiq &Sembiring, 2025).

The difficulty in understanding the concept of looping affects students' problem-solving abilities, resulting in lower performance (Cong & Ironsi, 2025; Pankiewicz & Baker, 2026; L. Wu et al., 2025; Xu et al., 2026). When solving simple problems such as finding the result of adding a series of numbers, students often make logical errors due to a lack of understanding of the iteration flow. Conventional teaching methods that rely solely on verbal explanations and code examples on the board or presentation slides are insufficient to help students build a clear mental representation of how the program runs. (Afandi, 2021; Hartono & Dermawan, 2021). As a result, learning becomes less effective, and students' understanding of looping concepts remains suboptimal. To address this issue, innovative learning media are needed to bridge the gap between theory and practice. One potential solution is the development of a Code Visualization Tool that can visualize the code execution process step by step. Through this visualization, us, the nd.

Corresponding Author:-Sulidar Fitri.
Address**:-**Postgraduate, Universitas Muhammadiyah Malang, Malang, Indonesia.

Several previous studies have shown that program visualization can enhance the understanding of fundamental programming concepts, especially for beginners. (Čisar et al., 2011; Lin et al., 2025; Maula et al., 2024; Monalisa et al., 2025; Parikesit& Amrullah, 2025; Schoenherr et al., 2024). Tools such as Python Tutor or algorithm animation have been widely used to help students trace program flow (Guo, 2013; Hundhausen & Brown, 2008; Mshvidobadze, 2021). However, most of these tools are still general in nature and do not specifically focus on looping material. In addition, student interactivity in setting inputs, modifying conditions, or stopping execution at

certain points remains limited. Therefore, it is necessary to develop a more contextual, dynamic, and learning-oriented visualization tailored to the needs of looping instruction.

Based on the above explanation, this study aims to develop a Code Visualization Tool specifically designed to enhance students' learning outcomes on the looping concept. This tool is expected not only to display the code execution flow but also to provide flexibility for students to conduct independent exploration, such as changing the initial variable values or looping conditions, thereby making the learning experience more active and in-depth. With this innovation, it is expected that the programming learning process will become more effective, engaging, and capable of improving students' learning outcomes, particularly in for and while loop commands.

Materials and Methods: -

Research Design

This study employed a Research and Development (R&D) method aimed at producing a web-based interactive learning media called Visual Code Tools to enhance students' learning outcomes on for and while looping concepts in programming. The development process followed the ADDIE model, which consists of five stages: Analysis, Design, Development, Implementation, and Evaluation (Branch, 2009).

The analysis stage aimed to identify the learning needs related to programming loop materials. A needs analysis was conducted by distributing questionnaires to students of the Fundamentals of Programming course. The questionnaire included aspects such as students' perceptions of programming learning and interaction toward visual code. Students often encounter difficulties in understanding the logical flow of looping structures such as for and while. These concepts, while fundamental in programming, can be abstract and challenging to grasp without concrete visualization.

The design stage focused on creating a visualization of looping flow. The steps in this stage included: 1) The learning material focuses on looping statements (*for, while, break, continue*). 2) Writing simple code in the editor. 3) Running a step-by-step visualization (Next/Run). 4) Observing variable. In the User Interface/ User experience (UI/UX) Design need the main components of the HTML version are: 1) Editor Area – a *<textarea>* element for writing pseudo-code. 2) Control Buttons – Next, Run, Pause, and Reset for controlling the simulation. 3) Output Console – displays the results of PRINT statements and execution logs.4) Source Viewer – highlights the active line using the *.highlight* class.

During the development stage, the visual code tools prototype was developed based on Designed as a single file in .html file tool that can run directly in a browser without installation or a server. Detects lines such as FOR, WHILE, END, PRINT, BREAK, CONTINUE, and assignments. Execute instructions step by step. Stores variable values and loop states in the program's internal state.

The product underwent validation by subject-matter experts and programming experts to evaluate content accuracy, design consistency, and interactivity. Expert feedback was used to refine the product. A small-scale trial was then conducted with a group of students to assess usability, clarity, and attractiveness. The results of this trial were analyzed and used to improve the product before conducting a large-scale implementation.

The implementation stage involved a field test with a group of students in class which major is Information Technology Education, especially in Universitas Muhammadiyah Tasikmalaya. The students who can participate are the ones that already contract the subject containing Programming language material. This phase included a pretest and post-test to measure students' learning improvement after using the visual code tools. The pretest measured students' initial understanding, while the post-test assessed learning outcomes after the intervention. The test results were analyzed using N-Gain score analysis to determine the effectiveness of the developed media in improving students' learning outcomes.

The evaluation stage aimed to revise and improve the visual code tools based on the weaknesses identified during the implementation stage. The final version of the product was refined and declared suitable for use as an interactive learning medium that can be implemented in programming courses.

Research Subjects and Objects

The population of this study consisted of all students enrolled in the Fundamentals of Programming course within the Information Technology Education program in Universitas Muhammadiyah Tasikmalaya. Sampling was conducted using the stratified random sampling technique to represent different levels of student ability. In the effectiveness testing phase, a simple random sampling technique was applied to minimize data bias. A total of 20 students were selected as respondents for the product effectiveness test.

Data collection technique

The data were collected through observation, questionnaires, and documentation. Observations were conducted to examine the initial learning conditions and identify difficulties in understanding the looping materials. Questionnaires were used to gather data from material, programming experts and students regarding the validity and usability of the product. The feasibility of the product was assessed using a Likert scale, and the quantitative data obtained were interpreted qualitatively to support product revision and refinement.

Data Analysis Techniques Data Analysis Techniques

Data analysis aimed to determine the effectiveness of the visual code tools in improving students' learning outcomes on for and while looping topics. The research employed a one-group pretest–post-test design. Instruments used included learning achievement tests and student perception questionnaires. The learning outcome data were analysed using N-Gain score analysis. The results were then interpreted to evaluate the effectiveness of the developed media in enhancing students' understanding and performance in programming.

$$N - Gain = \frac{PostTest\ Score - PreTest\ Score}{maximum\ score - pretest\ score}$$

Results and Discussion:-

Results

Many students struggle to mentally trace how each line of code is executed, how variable values change during each iteration, and at what point the loop stops or repeats. As a result, they frequently experience confusion when predicting program behaviour, identifying termination conditions, or debugging looping errors. This highlights the need for a visual learning tool that can represent loop execution step by step, allowing students to observe how control flow and variable states evolve throughout the looping process. In this study, the looping problem used is to calculate sum of numbers from 1 to 5 as shown in figure 1.

```
LET sum = 0
FOR i = 1 TO 5
  sum = sum + i
END
PRINT sum
```

```
LET i = 1
LET sum = 0

WHILE i <= 5
  sum = sum + i
  i = i + 1
END

PRINT sum
```

Figure 1. Pseudocode uses for and while loops to sum numbers from 1 to 5.

The display of this visual code tool as in figure 2 presents an interactive learning tool designed to help students understand the fundamental concepts of looping in programming, particularly the for and while structures. The main section contains an editor area where users can write or modify pseudo-code to be visualized. Below the editor are control buttons such Next, Run, Pause, and Reset which allow users to interactively manage the simulation process. Each time the Next button is pressed, the currently executed line of code is highlighted, enabling users to observe the program's execution flow in real time. This offers a learning experience like debugging, allowing students to see when loop conditions are tested, when variables change, and when the loop terminates.

At the bottom of the interface, there are two main panels: the Output Console and the Source Viewer. The Output Console displays the results of PRINT commands and logs the step-by-step execution activities, while the Source Viewer shows the complete code with the active line highlighted for easy tracking of the program's logic. Through this visualization, students can clearly observe how variables such as i and sum change during each iteration and when the loop stops according to the specified condition. Thus, this HTML-based Code Visual Tool not only facilitates theoretical understanding of looping concepts but also provides a more concrete, interactive, and intuitive learning experience in exploring algorithmic logic flow.

Table 1 presents the results of the material validation test conducted by the expert validator on the developed product. Based on the assessment results, the Material Alignment aspect obtained a score of 13 out of a maximum of 15, equivalent to 86.67%, indicating that the material is aligned with the intended learning objectives and expected competencies. Overall, the validation results show that the product falls into the category of feasible for use with minor revisions, as the average evaluation percentage exceeds 70%. After a thorough discussion and clarification session with the validator regarding the suggested revisions, the product's evaluation scores improved, resulting in an average increase of 1 point for each evaluation item.

The results of the programming validation test conducted by the expert validator on the technical aspects of the product are presented in Table 2. The assessment was carried out in two validation stages to ensure the overall quality of the code and system functionality. The Performance and Efficiency aspect received the highest score of 12 out of 15 (80%). After refining the program code, there was an improvement in the expert validator's evaluation scores for several items, indicating that the program now operates efficiently in terms of both execution speed and resource utilization.

Figure 2. Program output showing the visualized flow process of for and while loops.

Table 1. Results of Material Validation Test

No	Evaluation	Scores 1	Percentage	Scores 2	Percentage
1	Material Alignment	13	86,67 %	14	93,34%
2	Accuracy and Logical Correctness	11	73,34 %	13	86,67%
3	Learning Relevance	11	73,34 %	13	86,67%
4	Output Completeness	11	73,34 %	12	80%

Table 2. Results of Programmer Validation Test

No	Evaluation	Scores 1	Percentage	Scores 2	Percentage
1	Code Structure and Readability	10	66,67 %	12	80%
2	Algorithms and Logic Accuracy	15	100 %	15	100%
3	Performance and Efficiency	12	80 %	14	93,34%
4	User Interaction and Responsiveness	11	73,34 %	12	80%

Table 3. Results of Student Response

No	Evaluation	Scores	Maximum Scores	Percentage
1	Visual attraction	331	400	82,75 %
2	Usability	320	400	80 %
3	Learning Effectiveness	314	400	78,5 %
4	Engagement & Motivation	494	600	82,33 %

Table 3 presents the results of the student response evaluation toward the use of the Code Visual Tool as a learning medium. The findings show that the tool received positive feedback across all assessed aspects. The Visual Attraction aspect achieved a percentage of 82.75%, indicating that students found the interface visually appealing and easy to understand. The Usability aspect scored 80%, showing that the tool was considered practical and user-friendly during the learning process.

Meanwhile, the Learning Effectiveness aspect obtained 78.5%, suggesting that the tool effectively helped students comprehend programming loop concepts through interactive visualization. The Engagement and Motivation aspect achieved the highest score of 82.33%, demonstrating that the tool successfully increased student interest and motivation in learning programming. Overall, the results indicate that the Code Visual Tool is well received by students and feasible to use as an interactive learning aid for programming courses.

The boxplot in Figure 3 illustrates the comparison between the distribution of pre-test and post-test scores of the participants. Overall, there is a clear improvement in scores after the learning intervention. The range of pre-test scores appears wider, spanning from low (10) to high (100), indicating a considerable variation in participants' initial abilities. In contrast, the post-test scores are concentrated in the higher range (60 to 100), suggesting enhanced understanding and greater consistency in learning outcomes following the instructional process.

Furthermore, the position of the "X" mark representing the mean has shifted upward in the post-test group, providing additional evidence that the learning activity had a positive impact on participants' competency improvement. Thus, the boxplot demonstrates that the applied learning intervention effectively enhanced students' performance, both in terms of the average score and the distribution pattern, as further detailed in Table 4.

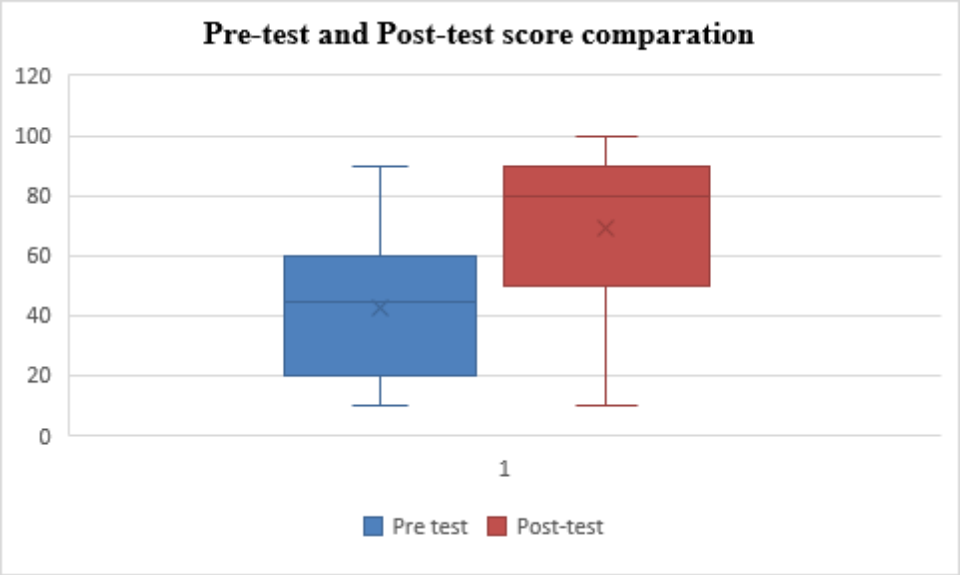

Figure 3. Comparison of Pre-test and Post-test score

Table 4. Frequency distribution for student score

Interval Score	Pre-test	Post-test
0 - 20	6	1
21 - 40	4	3
41 - 60	8	3
61 - 80	1	6
81 - 100	1	7

Table 5. Student Pre-test and Post-test assignment statistics

No	Evaluation	Learning Outcomes	
		Pre-test	Post-test
1	Number of Subjects	20	20
2	Highest Score	90	100
3	Lowest Score	10	10
4	Average	42,5	70

N-Gain= ((70-42,5))/((100-42,5)) = 27,5/57,5 = 0,478 or 0,48

The average pre-test score in table 5 was 42.5, which increased to 70 in the post-test. This improvement indicates a noticeable difference in learning outcomes before and after the intervention. According to Hake's (1998) classification, the N-Gain value of 0.48 falls into the "medium gain" category. This means that the implementation of the developed learning media or product was moderately effective in improving participants' learning outcomes.

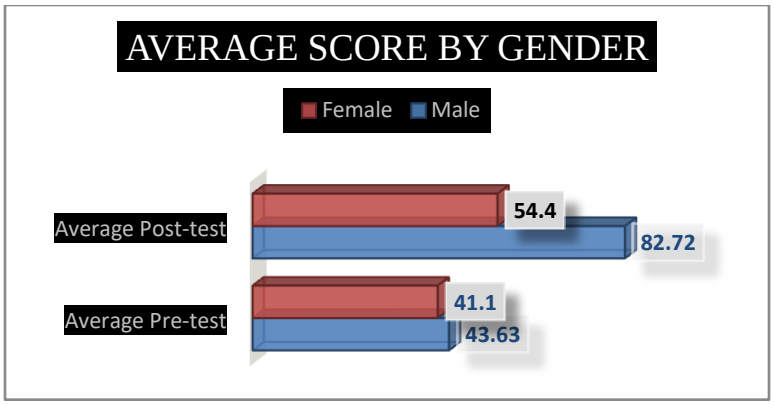

Figure 4. Comparison of average pre-test and post-test scores based on gender

Figure 4 shows that both male and female students experienced an improvement in their average scores after participating in the learning activities using the Code Visual Tool. Male students' average score increased from 43.63 in the pre-test to 82.72 in the post-test, while female students' scores rose from 41.1 to 54.4. Although both groups demonstrated progress, the improvement among male students was more significant, indicating that they may have benefited more effectively from the interactive visualization features of the learning media. Overall, the figure highlights that the implementation of the Code Visual Tool had a positive impact on both genders, contributing to an overall enhancement in programming comprehension.

Discussion: -

The results demonstrate that the Code Visualization Tool effectively enhances conceptual understanding of for and while loops. Visualization allows students to see the dynamic execution flow of each statement, bridging the gap between abstract programming syntax and concrete procedural logic. This aligns with findings by Minjie Hu et al. (2021) that visual learning media can improve code comprehension and reduce cognitive load for novice programmers (M. Hu et al., 2021).Furthermore, the tool's interactive simulation encourages active exploration. Students can modify loop parameters, observe changes in real time, and thus construct their own mental model of iteration. This constructivist approach supports deeper learning and aligns with the principle that active engagement leads to better retention (Wang et al., 2025).

Compared with traditional methods relying solely on static code examples or lecturer explanations, the visual code tool offers immediate feedback and tangible representation of logic flow. In conventional learning, students often struggle to grasp how loop variables change across iterations (Giannakoulas&Xinogalos, 2024; Lu & Hu, 2025); the visualization mitigates this difficulty by explicitly animating each execution step. The improvement in learning outcomes observed in this study reflects the advantages of visual interactivity in developing programming literacy, particularly in topics involving control flow and repetition structure.

The development of this tool contributes to the growing need for digital learning media that supports computational thinking (Salido et al., 2025). As higher education increasingly integrates programming across disciplines, tools that translate abstract logic into visual experience become crucial (Potocan et al., 2025). The Code Visualization Tool not only serves as a learning aid but also as a pedagogical innovation promoting self-directed and inquiry-based learning.

Although the tool proved effective, several limitations should be acknowledged. The implementation involved a limited number of participants and focused only on for and while loops. Future research could extend the tool's scope to include nested loops, conditional structures, and function calls, and integrate adaptive feedback features based on student performance analytics. Moreover, long-term studies are needed to assess retention effects and transfer of learning to more complex programming contexts.

Conclusion: -

This research concludes that the developed Code Visualization Tool successfully produced an effective interactive learning medium to enhance students' understanding of looping concepts in programming. Expert validation indicated that aspects such as material alignment, logical accuracy, and program performance met the feasibility standards with only minor revisions required. Field trials also demonstrated a significant improvement in students' learning outcomes, with an average score increase of 27.5 points and an N-Gain value of 0.48 (medium category).

Visualization, which presents the step-by-step execution of code, effectively helps students build a more concrete understanding of iteration logic and control structures. Moreover, the interactive features such as Next, Run, and Reset encourage independent exploration and strengthen active engagement during the learning process.Authors Contribution: Sulidar Fitri: conceptualization of the research, development of the methodology, data collection, and drafting of the manuscript. Eko Susetyarini: supervising the research process, validating results, conducting formal analysis, and reviewing and editing the manuscript. Elly Purwati: reviewing data processing and editing the manuscript.

References: -

Afandi, K. (2021). Identifikasi Proses yang MempengaruhiKemampuanPemrogramanMahasiswa Baru tanpaPengalaman Pemrograman. JATISI (Jurnal Teknik Informatika dan SistemInformasi), 8(4), 1785–1795. https://doi.org/10.35957/jatisi.v8i4.1120

Branch, R. M. (2009). Instructional Design: The ADDIE Approach. Springer Science & Business Media.https://doi.org/10.1007/978-0-387-09506-6_7

Čisar, S. M., Pinter, R., & Radosav, D. (2011). Effectiveness of Program Visualization in Learning Java: A Case Study with Jeliot 3. INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS &CONTROL, 6(4), 668–680.https://doi.org/10.15837/ijccc.2011.4.2094

Cong, L., & Ironsi, C. S. (2025). Integrating mobile learning and problem-based learning in improving students action competence in problem-solving and critical thinking skills. Humanities and Social Sciences Communications, 12(1), 1238. https://doi.org/10.1057/s41599-025-05397-4

Fitri, S., Susetyarini, E., Purwanti, E., Suryani Harahap, F., KurniatyRukman, N., Infatsirin, I., & Lubis, M. (2025). Mapping the role of tiktok in academic motivation: A bibliometric analysis of university students learning engagement. International Journal of Advanced Research, 13(05), 32–40. https://doi.org/10.21474/IJAR01/20873

Giannakoulas, A., &Xinogalos, S. (2024). A Critical Review of Primary School Students' Difficulties in Learning Programming Through Educational Games. Journal of Educational Computing Research, 63(3), 627–661. https://doi.org/10.1177/07356331241309074

Guo, P. J. (2013). Online python tutor: Embeddable web-based program visualization for cs education. 579–584. https://doi.org/10.1145/2445196.2445368

H. C. B. Chan, T. -L. Wong, P. Lohana, S. Mitra, & J. So. (2024). Top 10 Computer Science Open Educational Resources in MERLOT. 2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC), 2228–2232. https://doi.org/10.1109/COMPSAC61105.2024.00357

Hartono, B., & Dermawan, D. A. (2021). STUDI LITERATUR PENINGKATAN KEMAMPUAN BELAJAR SISWA MATA PELAJARAN PEMROGRAMAN DASAR DI SMK MENGGUNAKAN MODEL PEMBELAJARAN PROBLEM BASED LEARNING. Jurnal IT-EDU, 06(02). https://doi.org/10.26740/it-edu.v6i2.43426

Hundhausen, C. D., & Brown, J. L. (2008). Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study. Computers & Education, 50(1), 301–326. https://doi.org/10.1016/j.compedu.2006.06.002

Izul, I. A. M., Fitri, S., & Muhammad, T. (2024). Pengembangan Media PembelajaranPemrograman Materi Dasar Bahasa C Sharp Menggunakan Video Animasi: Development of Sharp C Language Basic Material Programming Learning Media Using Animated Videos. PRODUKTIF: JurnalIlmiah Pendidikan TeknologiInformasi, 8(1), 701–713.https://doi.org/10.35568/produktif.v8i1.4585

Lin, P.-W., Yu, S.-H., & Lai, C.-H. (2025). Dynamic Program Analysis and Visualized Learning System in University Programming Courses. Engineering Proceedings, 98(1), 30. https://doi.org/10.3390/engproc2025098030

Lu, M., & Hu, Z. (2025). Digital Twin-Enhanced Programming Education: An Empirical Study on Learning Engagement and Skill Acquisition. Computers, 14(8), 322. https://doi.org/10.3390/computers14080322

M. Hu, T. Assadi, & H. Mahroeian. (2021). Teaching Visualization-first for Novices to Understand Programming. 2021 IEEE International Conference on Engineering, Technology & Education (TALE), 654–660. https://doi.org/10.1109/TALE52509.2021.9678922

Manorat, P., Tuarob, S., &Pongpaichet, S. (2025). Artificial intelligence in computer programming education: A systematic literature review. Computers and Education: Artificial Intelligence, 8, 100403. https://doi.org/10.1016/j.caeai.2025.100403

Maula, N. F., Suryadi, D., &Jupri, A. (2024). Learning obstacles in the generalization process: In case number pattern topic. Research and Development in Education (RaDEn), 4(2), 964–976. https://doi.org/10.22219/raden.v4i2.36342

Monalisa, L. A., Kristiana, A. I., Fatahillah, A., Wihardjo, E., Aulia, V. A., Hussen, S., &Falebita, O. S. (2025). Development of interactive learning media assisted by scratch to enhance understanding of rectangle and square concepts. Research and Development in Education (RaDEn), 5(1), 741–753. https://doi.org/10.22219/raden.v5i1.40008

Mshvidobadze, T. (2021). Python for Automating Machine Learning Tasks. JINAV: Journal of Information and Visualization, 2(2), 77–82. https://doi.org/10.35877/454RI.jinav373

Pankiewicz, M., & Baker, R. S. (2026). Enhancing Student Focus and Problem-Solving with Real-Time LLM Feedback on Compiler Errors. In K. Tammets, S. Sosnovsky, R. Ferreira Mello, G. Pishtari, & T. Nazaretsky (Eds.), Two Decades of TEL. From Lessons Learnt to Challenges Ahead. EC-TEL 2025 (Vol. 16063). Springer. https://doi.org/10.1007/978-3-032-03870-8_28

Parikesit, B., & Amrullah, M. A. (2025). Development of Interactive Learning Media with InShot Application in Arabic Language Subjects. Arabiyatuna: Jurnal Bahasa Arab, 9(1), 149–164. https://doi.org/10.29240/jba.v9i1.11771

Permatasari, L., Yuana, R. A., & Maryano, D. (2018). Pemanfaatan Programming Assistance Tool Sebagai Upaya MeningkatkanMotivasi dan Hasil BelajarSiswa pada Kompetensi Dasar StrukturKontrolPerulangandalam Materi Pemrograman Dasar. Prosiding Seminar Nasional UNS Vocational Day, 1(0). https://doi.org/10.20961/uvd.v1i0.15964

Potocan, V., Nedelko, Z., & Rosi, M. (2025). Digitalization of Higher Education: Students' Perspectives. Education Sciences, 15(7), 847. https://doi.org/10.3390/educsci15070847

Salido, A., Syarif, I., Sitepu, M. S., Suparjan, Wana, P. R., Taufika, R., & Melisa, R. (2025). Integrating critical thinking and artificial intelligence in higher education: A bibliometric and systematic review of skills and strategies. Social Sciences & Humanities Open, 12, 101924. https://doi.org/10.1016/j.ssaho.2025.101924

Schoenherr, J., Strohmaier, A. R., &Schukajlow, S. (2024). Learning with visualizations helps: A meta-analysis of visualization interventions in mathematics education. Educational Research Review, 45, 100639. https://doi.org/10.1016/j.edurev.2024.100639

Syafiq, D. A., &Sembiring, Y. (2025). Pengembangan Model Visualisasi Data KependudukanBerbasisTeknologiInformasi untuk MendukungPengambilan Keputusan di Kecamatan Medan Amplas. JurnalPenelitianInovatif, 5(2), 1329–1338. https://doi.org/10.54082/jupin.1500

Wang, D., Dong, X., & Zhong, J. (2025). Enhance College AI Course Learning Experience with Constructivism-Based Blog Assignments. Education Sciences, 15(2), 217. https://doi.org/10.3390/educsci15020217

Wu, L., Xiang, X., Yang, X., Jin, X., Chen, L., & Liu, Q. (2025). Using hidden Markov model to detect problem-solving strategies in an interactive programming environment. Educational Technology Research and Development, 73(4), 2113–2130. https://doi.org/10.1007/s11423-025-10506-w

Wu, Z., & Wan, S. (2025). A Knowledge-Driven Approach to AI-Based Personalized Test Paper Creation in Programming Education. International Journal of Knowledge Management (IJKM), 21(1), 1–21. https://doi.org/10.4018/IJKM.369825

Xu, Z., Zhang, K., & Sheng, V. S. (2026). Logic Error Localization in Student Programming Assignments Using Pseudocode and Graph Neural Networks. In M. Mahmud, M. Doborjeh, K. Wong, A. C. S. Leung, Z. Doborjeh, & M. Tanveer (Eds.), Neural Information Processing. ICONIP 2024 (Vol. 2297). Springer. https://doi.org/10.1007/978-981-96-7036-9_4

Yuricha, Y., & Phan, I. K. (2023). Effectiveness of blended learning implementation for algorithm and programming course. JurnalInovasiTeknologi Pendidikan, 10(1), 43–54. https://doi.org/10.21831/jitp.v10i1.54707