



Development of OOPify as a Visual Block Programming Environment for OOP Learning and Group-Level Practical Performance

Naufal Oktavian
(Universitas Pendidikan
Indonesia, Bandung, Indonesia)

Yogi Prasetyo
(Universitas Pendidikan
Indonesia, Bandung, Indonesia)

Muhammad Rafi Valliansyah ✉
(Universitas Pendidikan
Indonesia, Bandung, Indonesia)

Lala Septem Riza
(Universitas Pendidikan
Indonesia, Bandung, Indonesia)

OPEN ACCESS

ARTICLE HISTORY

Received: June 10, 2026

Revised: July 23, 2026

Accepted: August 21,
2026

KEYWORDS

Block programming;
Multimedia learning;
Object-Oriented
Programming;
Practical performance;
Psychomotor skills.

ABSTRACT

Purpose – Object-Oriented Programming (OOP) is challenging for students because its abstract concepts are difficult to translate into practical programming solutions. This study developed OOPify, a visual block-based learning environment for OOP practical activities, and examined changes in students' conceptual understanding and group-level practical performance.

Methods – The study employed a Research and Development approach using the ADDIE model and a one-group pretest-posttest design. Participants were 34 vocational high school students in West Java, Indonesia. Data were collected through OOP conceptual understanding tests, group-level practical performance observations, expert validation, and student response questionnaires.

Findings – OOPify was rated feasible (82.5%) by one expert. The Wilcoxon Signed-Rank Test indicated a significant improvement in OOP conceptual understanding ($p < .001$), with 70.59% of students achieving moderate-to-high N-Gain and an average N-Gain of 0.50. Group-level practical performance, covering psychomotor, cognitive, and collaborative indicators, averaged 85.22% across Meetings 2 and 3. However, these results represent preliminary observational evidence because formal inter-rater reliability was not assessed.

Research implications – Visual block-based learning environments may support OOP learning by reducing syntactic complexity and allowing students to focus on concepts and problem-solving. Generalizability is limited by the single-school context and absence of a control group.

Originality – This study integrates visual block programming, OOP practical activities, and group-level practical performance within a single learning environment.

Correspondence Author: ✉rafivalli261@upi.edu

To cite this article : Oktavian, N., Prasetyo, Y., Valliansyah, M. R., & Riza, L. S. (2026). Development of OOPify as a Visual Block Programming Environment for OOP Learning and Group-Level Practical Performance. *Journal of Deep Learning, Computer Vision and Digital Image Processing*, 4(3), 122–143. <https://doi.org/10.61255/decoding.v4i3.1634>

This is an open access article under the CC BY-SA license



INTRODUCTION

Object-Oriented Programming (OOP) remains a challenging topic in programming education because students are required not only to understand abstract concepts but also to apply them through practical programming activities. Concepts such as classes, objects, methods, inheritance, and encapsulation require learners to develop abstract thinking skills, construct mental models, and establish relationships between conceptual knowledge and program implementation. Consequently, novice programmers often encounter difficulties in understanding and applying OOP concepts effectively, as reported by Batiha et al. [1], Khudhur et al. [2], Efan et al. [3], and Ragonis and Shmallo [4]. Previous studies have further indicated that these difficulties are not solely caused by the complexity of OOP concepts themselves but are also influenced by high cognitive load, limited programming experience, and challenges in transforming conceptual understanding into executable programming solutions [1], Heinonen et al. [5], and Wai et al. [6]. Moreover, a recent systematic mapping study by Tsichouridis et al. [7] revealed that although numerous educational environments have been developed to support OOP learning, most of them primarily focus on conceptual visualization and theoretical understanding rather than supporting students in solving more complex programming problems through practical activities. These findings highlight the need for learning media capable of visualizing OOP concepts while simultaneously facilitating practical learning experiences that help students connect abstract concepts with program implementation in a meaningful manner.

This challenge is also reflected in vocational education. A pilot study conducted at SMK Negeri 4 Padalarang revealed that 76.5% of students experienced difficulties in writing OOP code correctly and sequentially, 67.6% frequently made mistakes while typing program code, and 58.8% reported that although they understood OOP concepts theoretically, they still struggled to implement them in programming tasks. Interviews with teachers further indicated that students tended to understand programming concepts better through practical activities than through theory-oriented instruction. These findings suggest the existence of a substantial gap between conceptual understanding and practical programming skills in OOP learning.

Psychomotor skills constitute an important aspect of programming education because students are expected not only to understand concepts but also to apply their knowledge through practical activities. In OOP learning, students must be able to design solutions, construct program structures, implement concepts such as classes, objects, methods, inheritance, and encapsulation, and evaluate the functionality of developed programs. These activities involve manipulation, coordination, communication, and product creation, all of which are associated with psychomotor skill development. In computing education specifically, assessing psychomotor competence is inherently complex because programming activities are largely virtual rather than physical, and existing frameworks note that programming-related skills often reflect a combination of technical, cognitive, and psychomotor elements rather than psychomotor ability alone [8]. In this study, these three constructs are treated as conceptually distinct: OOP conceptual understanding refers to students' theoretical grasp of OOP concepts as measured through multiple-choice testing, whereas practical performance encompasses the broader psychomotor, cognitive, and collaborative behaviors observed during hands-on programming activities.

Research has shown that the development of psychomotor skills through structured learning activities contributes positively to learning outcomes and student engagement, as reported by Borrego-Balsalobre et al. [9]. Furthermore, advances in educational technology have demonstrated that interactive digital learning environments can support both the development and assessment of psychomotor skills through technology-enhanced practical activities, as noted by Barenji and Montreuil [10]. Therefore, OOP instruction should not focus exclusively on conceptual mastery but should also provide practical learning experiences that facilitate psychomotor skill development.

Among the approaches proposed to address these challenges, block-based visual programming environments have demonstrated considerable potential. Previous studies have shown that block-

based programming can help learners understand programming concepts through visual representations, reduce syntactic complexity, lower cognitive load, and improve programming achievement, particularly among novice learners, as reported by Hu et al. [11], Rodríguez-Martínez et al. [12], Hsu and Gainsburg [13], Saygıner et al. [14], and Yu et al. [15]. In addition, block-based environments support the transition from visual programming to text-based programming by simplifying syntax representation and visualizing program structures, allowing learners to focus more on logic development and problem solving than on syntax-related errors, as noted by Lin and Weintrop [16].

Within the context of OOP education, various educational programming environments have been developed to support students' understanding of abstract OOP concepts through visual representations of object relationships, program structures, and conceptual system designs [2], Chin et al. [17], and [7]. However, systematic literature reviews indicate that OOP education research is still dominated by studies focusing on instructional methods, teaching strategies, object-oriented pedagogical approaches, conceptual visualization, and learning outcome evaluation [3], [7], Gumonan et al. [18], Speth et al. [19], and Qu et al. [20]. Similarly, Gutiérrez et al. [21] reported that most solutions proposed for OOP education primarily address conceptual understanding and learning difficulties rather than integrating practical activities and comprehensive skill development. Consequently, there remains a need for learning environments that not only support conceptual understanding through practical activities but also facilitate the assessment of students' psychomotor skill development during the learning process.

The need for learning media capable of bridging conceptual understanding and practical skills has motivated the development of visual block-based programming environments for OOP instruction. One such environment is OOPify, a web-based multimedia learning platform that utilizes visual programming blocks to support students understand and implement OOP concepts. The development of digital learning environments such as OOPify aligns with findings reported by López-Pernas and Saqr [22], who argued that digital learning systems can serve not only as instructional delivery tools but also as sources of valuable information regarding learners' activities throughout the learning process.

Previous research by Huda et al. [23] introduced and implemented OOPify as a block-based programming medium for OOP learning to support learning evaluation and improve students' cognitive performance. The study reported that the visual representations provided by OOPify assisted students in understanding and implementing OOP concepts. Nevertheless, the previous work primarily focused on learning evaluation and cognitive achievement, while the use of OOPify as a practical learning medium that supports and analyzes observed group-level practical performance during OOP learning remains underexplored.

Therefore, the novelty of this study lies in the utilization of OOPify as a practical learning medium that integrates visual block programming activities with the analysis of students' psychomotor skills throughout the learning process. Through this approach, students are not only encouraged to construct object-oriented programming solutions visually but also to develop psychomotor skills through block manipulation, program construction, program testing, and case-study problem solving activities. Accordingly, this study aims to analyze changes in group-level psychomotor performance and students' Object-Oriented Programming competencies following the implementation of OOPify. The findings are expected to contribute to the development of more interactive programming learning media and to extend current knowledge regarding the integration of psychomotor skill development within visual block-based OOP learning environments.

METHOD

This study employed the Analyze, Design, Development, Implement, and Evaluate (ADDIE) instructional design model as the research and development framework. The research design used in this study was a Pre-Experimental Design in the form of a One-Group Pretest–Posttest Design. To provide a clear overview of the research procedure, a research framework was developed to illustrate all stages of the study, beginning with the preliminary study, followed by the development

phase, and ending with the documentation phase. During the development phase, activities focused on the development of OOPify as a learning medium to support students' psychomotor skills, as well as the development of psychomotor-oriented practical activities based on Student Worksheets (LKPD) that were implemented during the learning process. The complete research procedure is presented in Figure 1.

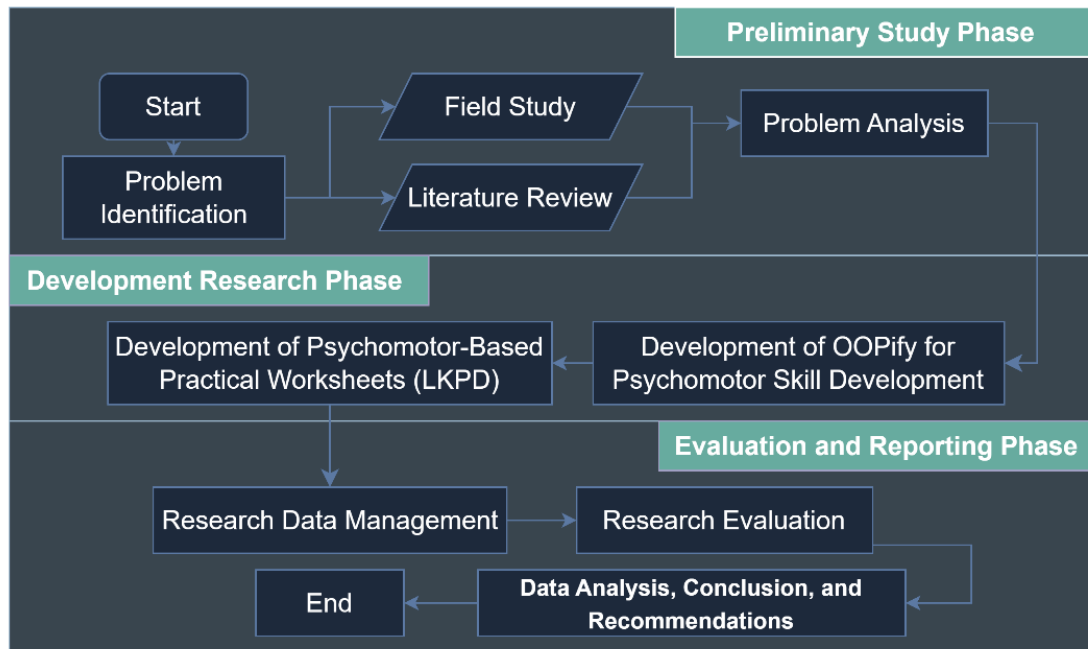


Figure 1. Research Procedure

As shown in Figure 1, the study began with a preliminary study consisting of problem identification, field study, and literature review to obtain information regarding the current conditions of OOP learning and the need for developing an appropriate learning medium. The results of this stage served as the basis for identifying and analyzing the problems addressed in this research. The study then proceeded to the development phase, which involved the development of OOPify as a multimedia learning environment to support OOP learning and students' psychomotor skills. In addition, psychomotor-oriented practical activities based on LKPD were designed and developed according to the intended OOP learning objectives. After the development process was completed, the study entered the evaluation and reporting phase, which included organizing and managing the collected research data, evaluating the research outcomes, conducting data analysis, and drawing conclusions and recommendations. All stages were carried out systematically to ensure the development of learning media and instructional materials that aligned with the objectives of the study.

Research Design

This study employed a Pre-Experimental One-Group Pretest–Posttest Design. In this design, students were first administered a pretest (O_1) to measure their initial OOP conceptual understanding. Subsequently, they participated in learning activities using the OOPify block-based multimedia learning environment (X). Finally, a posttest (O_2) was administered to examine changes in students' Object-Oriented Programming (OOP) skills following the implementation. Changes in students' OOP conceptual understanding were analyzed by comparing their pretest and posttest scores.

Population and Sample

The population of this study consisted of vocational high school students in Indonesia who studied Object-Oriented Programming (OOP). The sample was selected using a purposive sampling technique and comprised 34 eleventh-grade students from the Software and Game Development Program (*Pengembangan Perangkat Lunak dan Gim – PPLG*) at SMK Negeri 4 Padalarang, West Java,

Indonesia. The sample was chosen based on the relevance of the OOP curriculum and the school's readiness to support the research implementation. The selected sample represented one intact class used as the implementation group in the One-Group Pretest–Posttest Design. However, since the study was conducted in only one vocational high school in West Java, the findings have limited generalizability. Therefore, future studies involving larger and more diverse samples are recommended.

Research Instruments and Data Analysis

The research instruments included interviews and surveys conducted during the needs analysis phase, expert validation sheets, conceptual understanding tests, practical performance observation sheets, and student response questionnaires. Instrument validation in this study was conducted by a single expert: a lecturer from the Department of Computer Science Education, Universitas Pendidikan Indonesia, with expertise in Object-Oriented Programming and instructional media development. This expert validated both the conceptual understanding test items (pretest and posttest) and the OOPify multimedia environment, using the test-item validation sheet and the adapted General Programming Rubric, respectively. Because a single validator was involved, the validation results should be interpreted as an expert appraisal rather than as evidence of broad content or media validity; this limitation is further discussed in the Discussion section.

The conceptual understanding test was administered as both a pretest and posttest to examine changes in students' conceptual understanding before and after the implementation of OOPify. The pretest and posttest consisted of different sets of items. To ensure equivalence between the two test forms, both were developed based on the same learning objectives, content coverage, and cognitive levels specified in the test blueprint. The distribution of items across learning indicators and cognitive levels was designed to be comparable between the two forms. Each form measured the same Object-Oriented Programming concepts using comparable indicators and underwent the same expert validation process prior to implementation. Prior to implementation, both test forms were evaluated for internal consistency using Cronbach's Alpha reliability analysis. The pretest instrument obtained a Cronbach's Alpha coefficient of 0.870, while the posttest instrument obtained a coefficient of 0.844, indicating very high internal consistency for both test forms.

In addition to reliability analysis, item difficulty and discrimination indices were calculated using the same pilot-test data, involving 13 students for the pretest form and 13 students for the posttest form (26 students in total). Item difficulty indices were interpreted using standard criteria: 0.00–0.30 (difficult), 0.31–0.70 (medium), and 0.71–1.00 (easy) [24]. Discrimination indices were interpreted as poor ($D < 0.20$), fair ($0.20 \leq D < 0.30$), good ($0.30 \leq D < 0.40$), and very good ($D \geq 0.40$) [25]. As shown in Table 1, the two forms demonstrated nearly identical average difficulty indices ($P = 0.539$ for both forms) and consistently high discrimination indices ($D = 0.589$ for the pretest and 0.522 for the posttest). Taken together, the high reliability coefficients and the comparable difficulty and discrimination indices provide converging evidence that the pretest and posttest forms were both internally consistent and empirically equivalent in demanding level, supporting their use for measuring pre–post changes in students' conceptual understanding.

Table 1. Comparison of Item Difficulty and Discrimination Indices between Pretest and Posttest Forms

Indicator	Pretest (Package 1)	Posttest (Package 2)
Number of items analyzed	30	30
Average Difficulty Index (P)	0.539 (Medium)	0.539 (Medium)
Difficulty category distribution	27 Medium, 3 Easy	26 Medium, 3 Easy, 1 Difficult
Average Discrimination Index (D)	0.589 (Very Good)	0.522 (Very Good)
Discrimination category distribution	20 Very Good, 7 Good, 3 Poor	18 Very Good, 8 Good, 4 Poor

As shown in Table 1, the pretest and posttest forms exhibited nearly identical average difficulty indices, both falling within the medium category, and comparable average discrimination indices, both categorized as very good. Only one posttest item was classified as difficult, while the remaining items in both forms fell within the medium-to-easy range. These results indicate that the two test

forms were empirically comparable in terms of item difficulty and their ability to differentiate higher- and lower-performing students.

Practical performance observations were conducted during practical learning activities using OOPify. Psychomotor observations were not conducted during the first meeting because the activities focused on orientation, introduction to OOPify, and pretest administration. Therefore, psychomotor assessment was limited to the practical learning activities conducted in Meetings 2 and 3. Psychomotor observations were conducted by two observers, namely the researcher and a research assistant. Before the observations were conducted, both observers discussed the assessment criteria and agreed on the scoring procedures to promote consistency throughout the observation process. The 34 students participating in the study were organized into 11 groups consisting of three to four students per group. Group-level observation was employed because the practical activities required students to collaboratively discuss solutions, construct block-based programs, test program functionality, and complete case-study tasks. Therefore, psychomotor performance was assessed at the group level based on each group's observable actions and interactions during the learning process rather than on individual students' performance. The observation focused on four psychomotor dimensions adapted from Trowbridge et al. [26], namely moving, manipulating, communicating, and creating.

Student responses were measured using a usability-testing questionnaire developed based on Nielsen's five usability attributes [27]: learnability, efficiency, memorability, errors, and satisfaction. The questionnaire consisted of 10 items, with two items representing each dimension (see Table 2 for the complete list of items).

Table 2. Usability Questionnaire Items Based on Nielsen's Five Usability Attributes

No	Dimension	Item
1	Learnability	I could understand how to use OOPify with minimal explanation.
2	Learnability	I found it easy to learn the basic functions available in OOPify.
3	Efficiency	I was able to complete class and object creation tasks quickly using OOPify.
4	Efficiency	I did not waste time while learning with OOPify.
5	Memorability	After using OOPify several times, I could still remember how to use it without having to relearn it.
6	Memorability	I did not find it difficult to remember the steps when creating a project in OOPify.
7	Errors	I rarely made mistakes while using OOPify.
8	Errors	When I made a mistake, I could easily fix it in OOPify.
9	Satisfaction	I felt satisfied using OOPify as a learning medium for OOP.
10	Satisfaction	I would recommend OOPify to other students for learning OOP.

All items were positively phrased; therefore, no reverse scoring was applied prior to calculating the percentage scores. Media feasibility scores were interpreted as very feasible (81–100%), feasible (61–80%), moderately feasible (41–60%), less feasible (21–40%), or not feasible (0–20%) [28]. Student response scores were interpreted using the same percentage thresholds, categorized as very strong (81–100%), strong (61–80%), moderately strong (41–60%), weak (21–40%), or very weak (0–20%) agreement [29].

The collected data were analyzed using descriptive quantitative techniques. Expert validation results, practical performance, and student responses were analyzed using percentage scores. Practical performance scores were obtained from observations using a five-point Likert scale and then converted into percentages. Changes in students' OOP conceptual understanding were analyzed by comparing the pretest and posttest scores. The magnitude of the observed change was subsequently quantified using the Normalized Gain (N-Gain) formula. The normalized gain (N-Gain) was calculated using the formula proposed by Hake [30]:

$$g = \frac{\text{Posttest} - \text{Pretest}}{100 - \text{Pretest}}$$

The N-Gain values were classified into three categories, namely low ($g < 0.30$), medium ($0.30 \leq g < 0.70$), and high ($g \geq 0.70$), following the classification proposed by Hake [30]. Negative N-Gain values were retained in the analysis because they indicate that a student's posttest score was lower than the

corresponding pretest score. The average N-Gain was calculated as the arithmetic mean of individual students' N-Gain scores rather than from the class-level pretest and posttest mean scores.

Research Procedure

This study adopted the ADDIE instructional design model (*Analysis, Design, Development, Implementation, and Evaluation*) as the framework for developing the OOPify learning media. Through this model, each development stage was conducted systematically, beginning with needs analysis and ending with the evaluation of the implemented learning media. The development procedure employed in this study is illustrated in Figure 2.

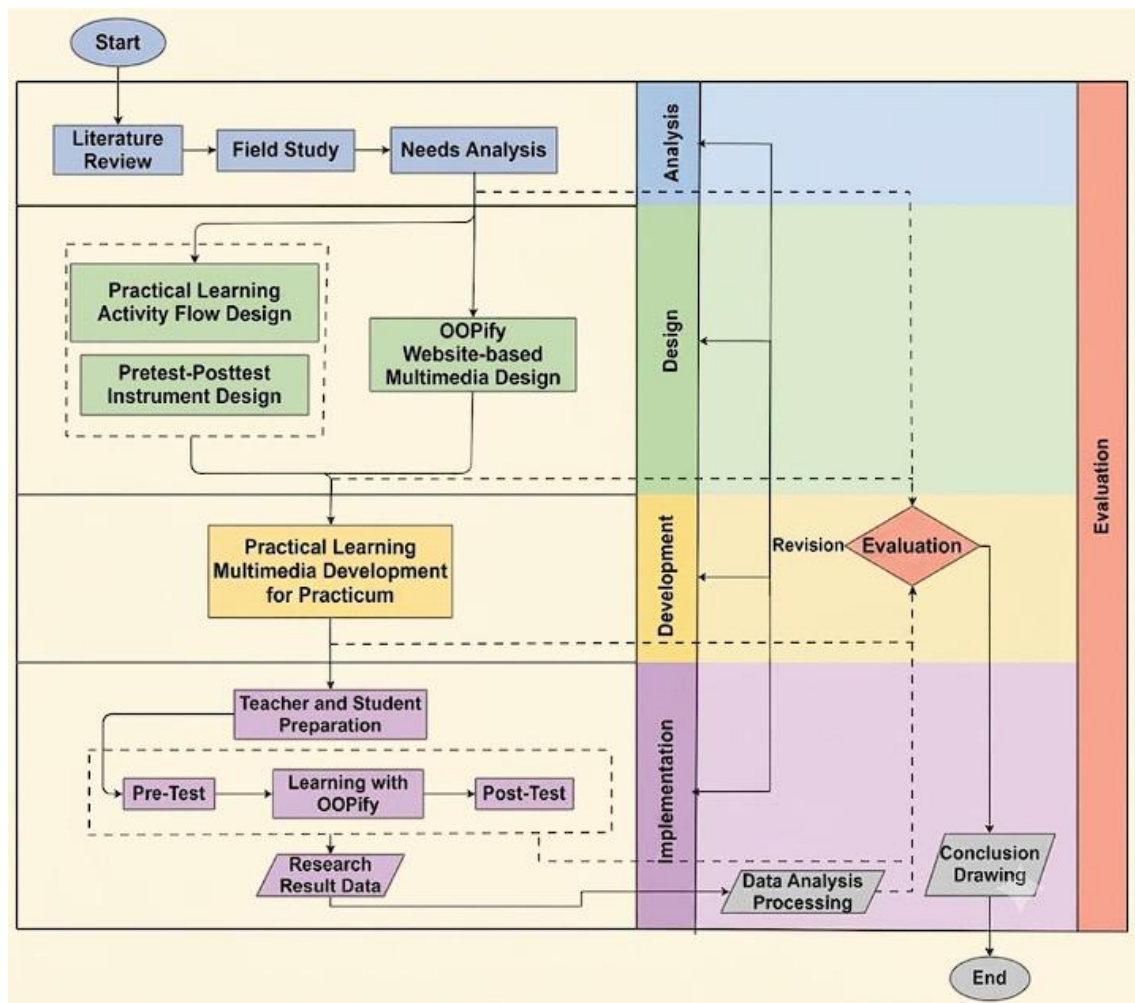


Figure 2. Development Procedure

In the Analysis stage, learning needs and students' difficulties in learning OOP were identified. The Design stage involved designing the OOPify media, developing research instruments, and planning learning activities. The Development stage involved developing the media and validating it through an expert with combined expertise in Object-Oriented Programming and instructional media. The Implementation stage involved applying OOPify in OOP practical learning activities with eleventh-grade vocational high school students. Finally, the Evaluation stage examined the feasibility of the developed media, changes in students' OOP conceptual understanding based on the pretest and posttest results, observed group-level practical performance during practical learning activities, and students' responses to the developed media.

a. Analysis Stage

At this stage, a needs analysis was conducted through classroom observations, interviews with Software and Game Development (PPLG) teachers, and student surveys to identify challenges in learning Object-Oriented Programming (OOP).

b. Design Stage

This stage included the development of storyboards, user interface (UI) design, preparation of learning materials, and the design of research instruments used during the development and evaluation processes.

c. Development Stage

During this stage, the OOPify block-based multimedia learning environment was developed according to the previously established design specifications. The developed multimedia was subsequently validated by a lecturer from the Department of Computer Science Education, Universitas Pendidikan Indonesia, who has expertise in instructional media development and educational technology. The validator was selected based on academic expertise and experience in developing and evaluating computer-based learning media. The media was evaluated using an adapted version of the General Programming Rubric proposed by Bailie et al. [31]. From the original rubric, four aspects relevant to multimedia learning evaluation were adopted: solution quality, program design, user interface, and code readability. Aspects related to source-code documentation and maintainability were excluded because they are more applicable to evaluating student-written software than to evaluating instructional multimedia. Each criterion was rated using a five-point Likert scale. The obtained scores were converted into percentages to facilitate interpretation of the overall media quality. Higher percentage scores indicate better compliance with the rubric criteria and greater media feasibility for classroom implementation. The media was revised prior to implementation based on the suggestions and recommendations provided by the validator to enhance its quality and suitability for classroom use.

d. Implementation Stage

OOPify was implemented with students in classroom learning activities. The implementation began with the administration of a pretest, followed by learning and practical activities using OOPify, and concluded with a posttest and the completion of student response questionnaires.

e. Evaluation Stage

The evaluation stage aimed to assess the quality of the developed media based on expert validation results, changes in students' OOP conceptual understanding as reflected by the pretest and posttest results, observed group-level practical performance during practical activities, and students' responses to the use of OOPify. To observe students' practical performance during practical activities using OOPify, an observation instrument was developed based on the psychomotor skill classification proposed by Trowbridge et al. [26]. The instrument consisted of four dimensions: moving, manipulating, communicating, and creating. These dimensions were used to describe students' abilities to perform physical actions, operate learning tools or media, communicate during the learning process, and produce products or artifacts. Because several indicators (e.g., asking questions, discussing tasks, completing a mini-project) also involve cognitive, collaborative, and engagement-related behaviors in addition to physical/motor actions, this instrument is more accurately described as an observational rubric for group-level practical performance rather than a pure measure of psychomotor competence. The details of the psychomotor dimensions, sub-indicators, and assessment codes are presented in Table 3.

Table 3. Observational Rubric for Group-Level Practical Performance (adapted from psychomotor indicators proposed by Trowbridge et al. [26])

No	Aspect	Practical Performance Indicator	Code
1.	Moving	Independently turns on the laptop/computer and accesses the OOPify platform.	Moving 1
		Prepares the required learning resources (notebook, digital worksheet, and coding notes) according to the practical activity instructions.	Moving 2
2.	Manipulating	Arranges class, object, and method blocks in OOPify according to the teacher's instructions.	Manipulating 1
			Manipulating 2

No	Aspect	Practical Performance Indicator	Code
		Creates methods and connects them to the appropriate objects in OOPify.	
3.	Communicating	Asks questions to the teacher when encountering difficulties in arranging or connecting blocks.	Communicating 1
		Engages in discussions with peers regarding tasks and activities in OOPify.	Communicating 2
4.	Creating	Completes a simple mini-project by applying Object-Oriented Programming principles using OOPify.	Creating 1
		Produces a digital practical report containing screenshots and descriptions of the developed block-based program.	Creating 2

Based on Table 3, the moving aspect was used to assess students' ability to perform initial preparations for practical activities and operate the required equipment. The manipulating aspect focused on students' skills in using OOPify features to construct and modify program blocks according to OOP concepts. The communicating aspect was used to observe students' interactions with teachers and peers during practical activities. Meanwhile, the creating aspect evaluated students' ability to produce outputs, such as simple projects and practical reports. These four dimensions were adapted from the psychomotor skill framework presented in [26] and were employed to analyze group-level psychomotor performance during Object-Oriented Programming learning activities using OOPify.

RESULTS AND DISCUSSION

Before conducting inferential statistical analyses, the pretest and posttest data were analyzed descriptively to provide an overview of students' Object-Oriented Programming (OOP) skills before and after participating in learning activities using OOPify. Descriptive statistical analysis was performed to examine the characteristics of the data distribution through measures of central tendency, dispersion, and distribution shape. The statistics analyzed included sample size, total score, mean, median, mode, quartiles, minimum value, maximum value, range, standard deviation, variance, skewness, and kurtosis. The results of the descriptive statistical analysis are presented in Table 4.

Table 4. Descriptive Statistics of Pretest and Posttest Scores

Statistics	Pretest	Posttest
Sample Size (N)	34	34
Total Score	1568	2480
Mean	46.12	72.94
Median (Q2)	44.00	76.00
Mode	44	76
Quartile 1 (Q1)	37.00	64.00
Quartile 3 (Q3)	56.00	87.00
Minimum	20	28
Maximum	80	100
Range	60	72
Standard Deviation	14.00	18.58
Variance	196.11	345.27
Skewness	0.595	-0.959
Kurtosis	0.251	0.314

Based on Table 4, the mean pretest score was 46.12 and increased to 72.94 in the posttest. Similar increases were observed in the median (44.00 to 76.00), mode (44 to 76), minimum score (20 to 28), and maximum score (80 to 100), indicating higher OOP conceptual understanding scores in the posttest than in the pretest. Furthermore, the quartile range shifted from 37–56 in the pretest to 64–87 in the posttest, suggesting that higher scores were observed across most students in the posttest. The skewness value changed from 0.595 to -0.959, indicating that the score distribution shifted toward higher achievement levels. Overall, the descriptive statistics indicate that students obtained higher OOP conceptual understanding scores in the posttest than in the pretest following the implementation of OOPify.

Analyze Stage Outcome

The analysis stage was conducted through teacher interviews, student questionnaires, and learning needs analysis related to Object-Oriented Programming (OOP) instruction at SMK Negeri 4 Padalarang. The findings revealed that students experienced difficulties in understanding fundamental OOP concepts, including classes, attributes, methods, and objects. In addition, students reported that they tended to understand OOP concepts better through practical activities than through theory-oriented instruction. Survey results showed that 76.5% of students experienced difficulties writing OOP code correctly, while 67.6% frequently encountered syntax errors. These findings indicated the need for a practice-oriented learning medium capable of visualizing OOP concepts and facilitating students' understanding. Based on these needs, OOPify was developed as a block-programming-based learning multimedia platform to support OOP practical activities.

Design Stage Outcome

To support the implementation of the study, a learning scenario consisting of three instructional meetings was designed using OOPify as a visual block-programming learning medium. Each meeting was structured with different learning activities aligned with the intended learning objectives, beginning with students' initial competency assessment, followed by instructional delivery and worksheet-based practical activities, and ending with the evaluation of learning outcomes. The allocation of time and sequence of activities for each meeting are presented in the Gantt chart shown in Figure 3.

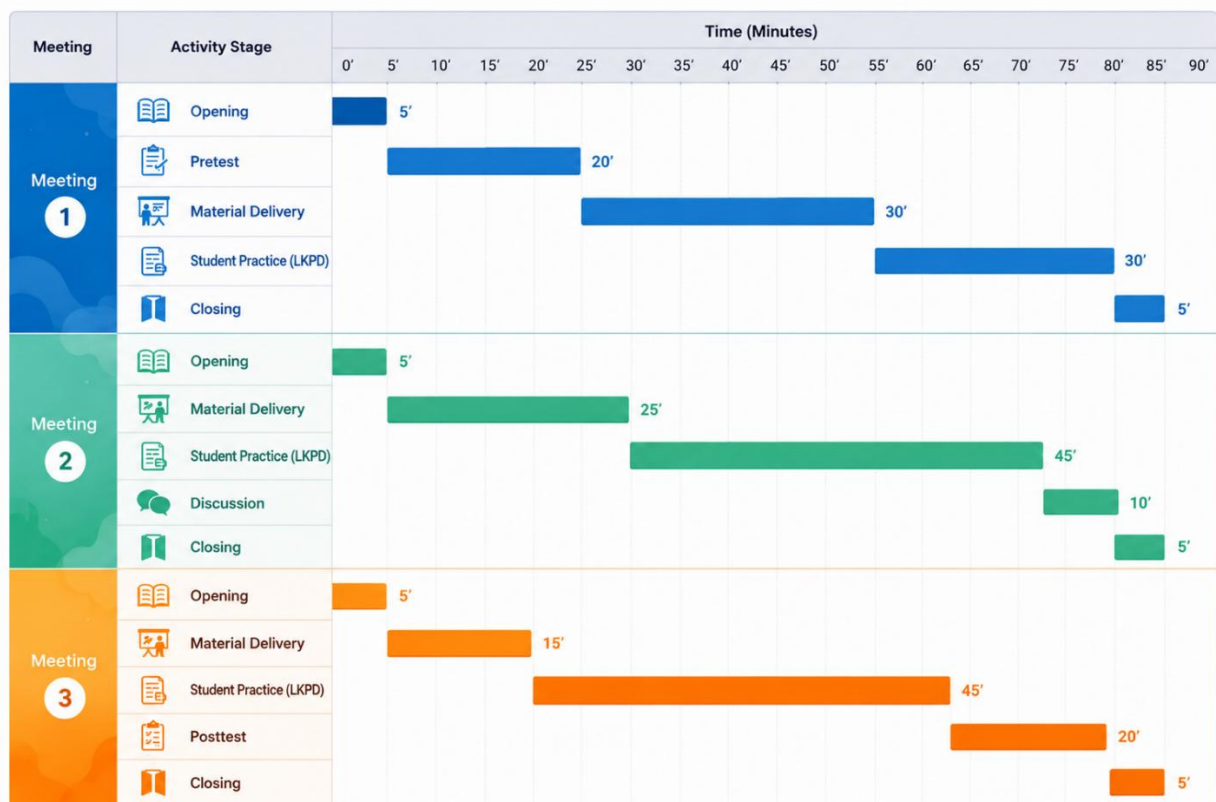


Figure 3. Gantt Chart of the Learning Scenario

As illustrated in Figure 3, the learning process was conducted across three meetings, each lasting 90 minutes. The first meeting focused on administering the pretest, introducing fundamental OOP concepts, and conducting program analysis activities using student worksheets. The second meeting emphasized concept reinforcement and practical activities using OOPify, complemented by discussion sessions to strengthen students' understanding. The third meeting focused on more independent practical activities guided by worksheets, followed by a posttest to examine changes in students' OOP conceptual understanding. This sequence was designed progressively to provide

students with systematic learning experiences in understanding and implementing OOP concepts using OOPify.

Development Stage Outcome

During the development stage, the designed prototype was implemented as OOPify, a web-based multimedia learning environment for Object-Oriented Programming practical activities. The platform was developed using HTML, CSS, and JavaScript to ensure interactivity, lightweight performance, and accessibility through web browsers. Figure 4 presents the main interface of OOPify along with an example of block assembly. The platform consists of course modules, a collection of OOP blocks, and an interactive workspace for practical programming activities. Through block programming, students can construct programs using blocks representing OOP concepts such as classes, attributes, methods, and encapsulation. OOPify also includes drag-and-drop, snapping, zooming, reset, and trash-bin functionalities to facilitate practical learning activities.

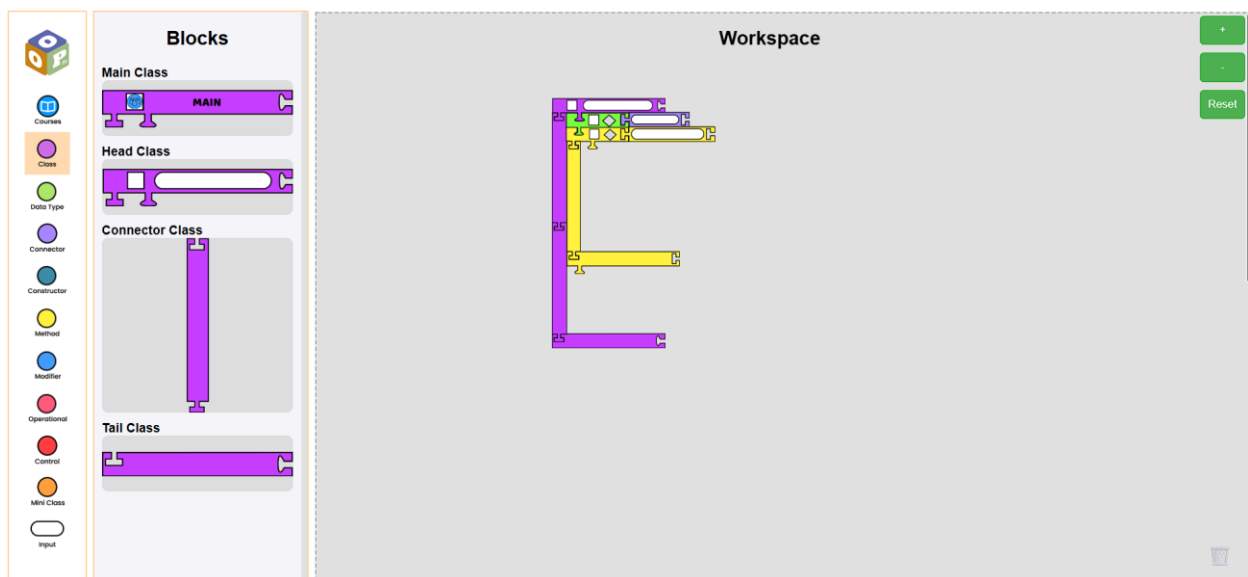


Figure 4. Main Interface of OOPify with an Example of Block Assembly

In addition to the multimedia platform, psychomotor-oriented student worksheets were developed and integrated with OOPify to guide students in constructing block-based programs systematically. These worksheets were designed to help students connect OOP concepts with their implementation during practical activities. After development, both the multimedia platform and worksheets were validated by the expert described in Section 2.3. The validation results indicated that OOPify was appropriate for use as a practical learning multimedia tool in OOP instruction.

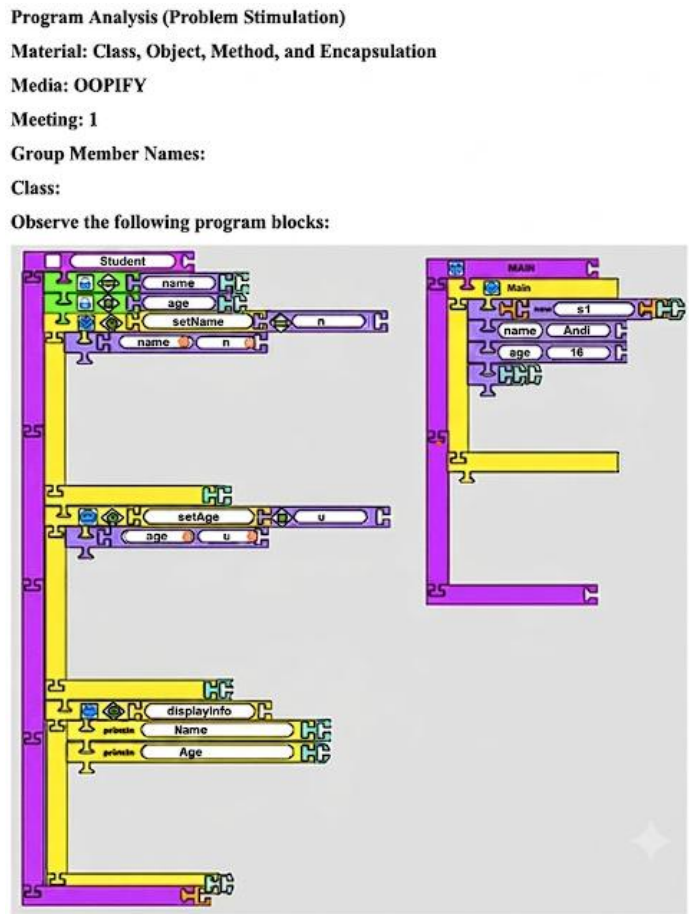


Figure 5. Example of Student Worksheet (Meeting 1)

As shown in Figure 5 and Table 5, students were required to identify a case study represented by a given program, determine three key pieces of information embedded within the program blocks, analyze the input-process-output (IPO) components, predict the program output, and formulate conclusions based on their analysis. These activities were designed to train students’ ability to understand program logic and connect visual program representations with OOP concepts before engaging in practical programming tasks.

Table 5. Activities Included in the Student Worksheet

No	Student Worksheet Questions
	In your opinion, what kind of scenario or story would be suitable for this program?
1.	Write your answer below:
	After observing the OOPify block program above, identify and explain the three most important elements shown in the program.
2.	Write your answer below:
	What are the input, process, and output of the program?
3.	Write your answer below:
	Write the output that will be displayed when the program is executed.
4.	Write your answer below:
	Write your conclusion based on the activity you have completed.
5.	Write your answer below:

Implementation Stage Outcome

The implementation stage involved 34 students. The instructional process began with a pretest to assess students' prior knowledge of OOP concepts. Subsequently, students participated in learning activities that integrated OOP instruction with practical exercises using OOPify. During the practical sessions, students constructed object-oriented programs using visual blocks representing classes, attributes, methods, and encapsulation. These activities were conducted progressively with guidance from the developed worksheets. Throughout the learning process, group-level psychomotor performance was observed using the prepared observation instrument. Upon completion of all practical activities, students were administered a posttest to examine changes in their OOP skills. Additionally, students completed a response questionnaire to evaluate their perceptions of OOPify as a practical learning multimedia tool. All collected data were subsequently analyzed during the evaluation stage to examine the feasibility of the developed media, changes in students' OOP conceptual understanding, observed group-level practical performance, and students' responses toward OOPify.

Evaluation Stage Outcome

The evaluation stage was conducted to evaluate the developed OOPify multimedia and the outcomes observed following its implementation. The evaluation focused on three aspects: observed group-level practical performance, changes in students' OOP conceptual understanding based on the pretest and posttest results, and students' responses toward the learning media.

1. Observed Group-Level Practical Performance

To examine the development of observed group-level practical performance (encompassing psychomotor, cognitive, and collaborative behaviors) during the use of OOPify, analyses were conducted for four psychomotor dimensions: moving, manipulating, communicating, and creating. The analysis was based on the cumulative observation scores of 11 student groups across instructional meetings using a five-point Likert scale. The progression of the observed group-level practical performance is illustrated in Figure 6.

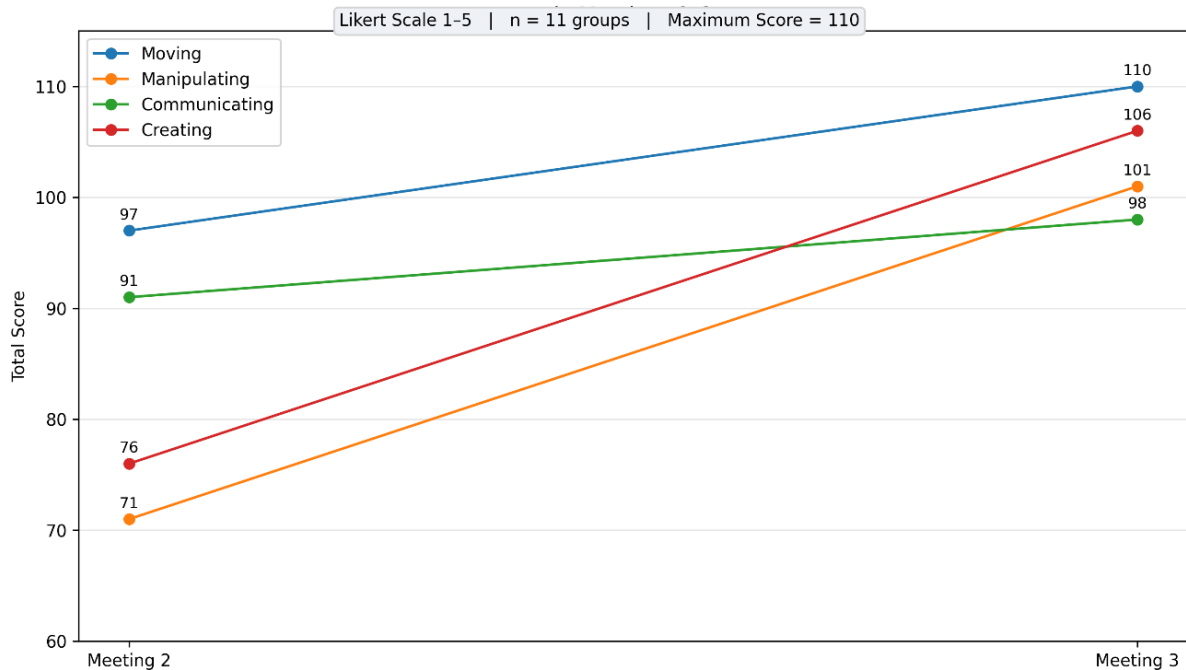


Figure 6. Line Chart of Students' Practical Performance Scores Across Meetings 2-3

Based on Figure 6, psychomotor observations were conducted only during the practical learning activities in Meetings 2 and 3. All four psychomotor dimensions showed higher observation scores in Meeting 3 than in Meeting 2. During the second meeting, the moving dimension obtained the highest score (97), followed by communicating (91), creating (76), and manipulating (71). In the

third meeting, the observation scores increased to 110 for moving, 101 for manipulating, 98 for communicating, and 106 for creating. These higher observation scores indicate that the observed student groups performed more successfully in operating the learning platform, constructing program blocks, communicating during practical activities, and completing OOP-based programming tasks during the second observation session. Overall, the findings indicate that the observed group-level practical performance scores were higher in Meeting 3 than in Meeting 2, with a combined average score across both meetings of 85.22%, categorized as very high. As these scores were derived from group-level classroom observations without formal inter-rater reliability analysis, they should be interpreted as preliminary observational evidence rather than definitive measurements of psychomotor competence.

2. Comparison of Pretest and Posttest Histograms

To provide a clearer picture of changes in students' Object-Oriented Programming (OOP) abilities before and after learning with OOPify, an analysis of the pretest and posttest score distributions was conducted. Distribution analysis not only reveals differences in average learning outcomes but also illustrates the patterns of score dispersion across the two measurements. Through data distribution visualization, it is possible to observe shifts in student performance after participating in learning and practical activities using the OOPify learning media. The distributions of pretest and posttest scores are presented in Figure 7.

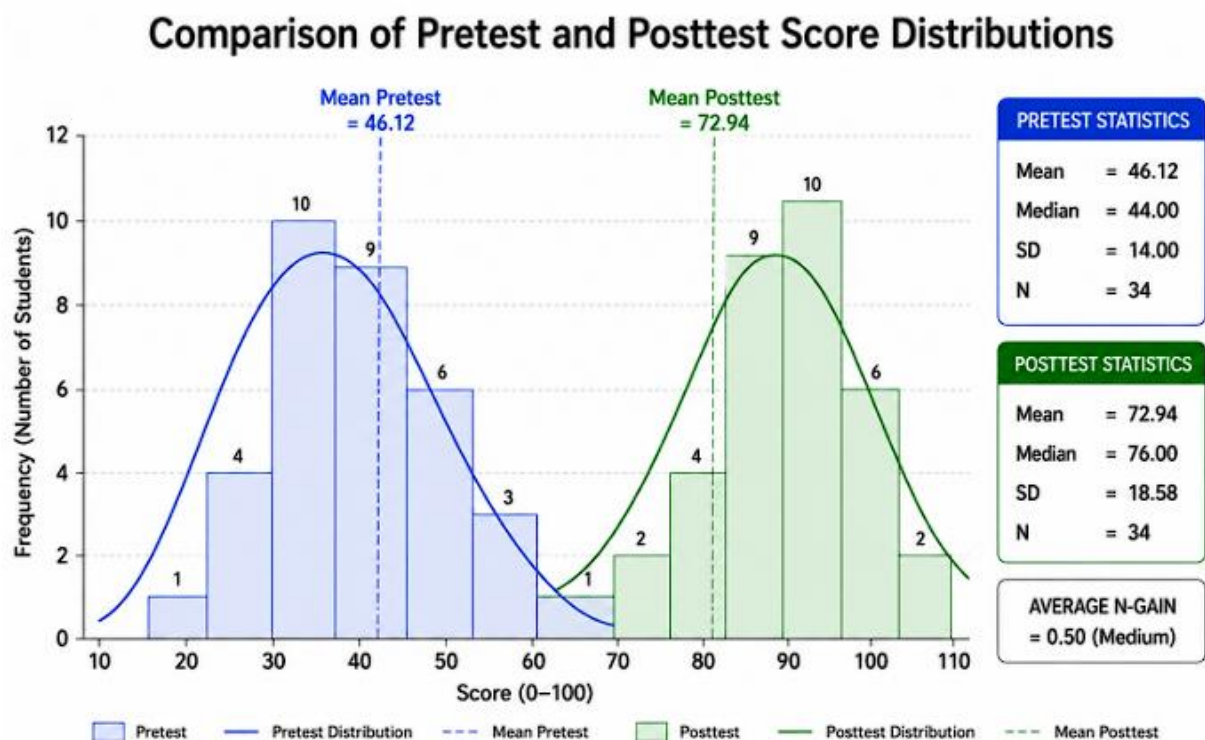


Figure 7. Distribution Histogram of Pretest and Posttest Scores

Based on Figure 7, the score distribution shifted toward higher scores in the posttest compared with the pretest. The mean score increased from 46.12 in the pretest to 72.94 in the posttest. Similarly, the median score increased from 44.00 to 76.00, indicating that the posttest scores were generally higher than the pre-test scores. The posttest distribution also exhibited a greater concentration of scores within higher score ranges, as reflected by the rightward shift of the distribution peak. Although the standard deviation of the posttest scores (18.58) was slightly higher than that of the pretest scores (14.00), the wider dispersion suggests greater variability in posttest performance among students. Overall, the shift of the posttest distribution toward higher scores indicates that higher OOP conceptual understanding scores were observed in the posttest than in the pretest following the implementation of the learning activities using OOPify.

3. Conceptual Understanding of OOP

Changes in students' OOP conceptual understanding were analyzed using the pretest and posttest results collected before and after the learning activities using OOPify. The analysis included normality testing, the Wilcoxon Signed-Rank Test, and N-Gain analysis to examine differences between the pretest and posttest scores and the distribution of individual learning gains. Normality was assessed using the Shapiro–Wilk test because the sample size consisted of 34 participants. The results indicated that the pretest scores were normally distributed ($W = 0.960$, $p = .237$), whereas the posttest scores were not normally distributed ($W = 0.912$, $p = .009$). Therefore, the Wilcoxon Signed-Rank Test was employed to compare the pretest and posttest scores. The test revealed a statistically significant difference between the pretest and posttest scores ($Z = -4.826$, $p < .001$). Among the 34 participants, 30 showed higher posttest scores than pretest scores, whereas four obtained lower posttest scores and no tied scores were observed. The corresponding effect size was large ($r = 0.83$), indicating a substantial difference between the pretest and posttest scores.

To further examine changes in students' OOP conceptual understanding scores, an analysis of the N-Gain distribution was performed using the pretest and posttest results. This analysis aimed to examine the distribution of learning gains across the low, medium, and high categories. In addition, the histogram was supplemented with a distribution curve and standard deviation values to illustrate the pattern of data dispersion around the mean. The distribution of students' N-Gain scores is presented in Figure 8.

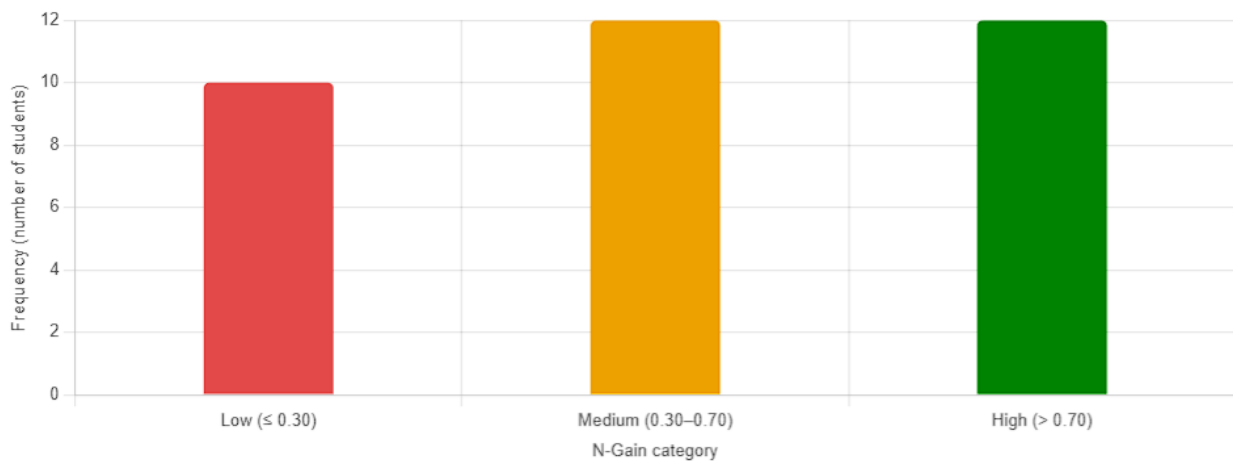


Figure 8. N-Gain Distribution Bar Chart

As shown in Figure 8 and Table 6, students' N-Gain scores were distributed across the low, medium, and high categories, with the medium and high categories accounting for the majority of students. Specifically, 24 out of 34 students (70.59%) were classified in the moderate-to-high N-Gain categories. The average N-Gain score was 0.50, corresponding to the moderate category according to Hake's classification.

Table 6. Distribution of Students Based on N-Gain Categories

Category	Frequency	Percentage
Low	10	29.41%
Medium	12	35.29%
High	12	35.29%
Total	34	100%

As shown in Table 6, 12 students (35.29%) were classified in the medium N-Gain category, while 12 students (35.29%) were classified in the high category. Therefore, 24 out of 34 students (70.59%) were classified in the moderate-to-high N-Gain categories. The remaining 10 students (29.41%) were classified in the low category, including four students who obtained negative N-Gain values because their posttest scores were lower than their corresponding pretest scores.

4. Material Validation

Assessment instrument validation was conducted through expert judgement by the expert described in Section 2.3 to evaluate the alignment of the test items with Bloom's taxonomy, the learning objectives, and the assessment indicators. The initial instrument consisted of 60 multiple-choice questions designed to measure students' conceptual understanding of Object-Oriented Programming (OOP). Based on the expert review, several items were revised or removed to improve their alignment with the intended learning objectives and assessment indicators. A summary of the expert validation results is presented in Table 7.

Table 7. Summary of Expert Validation Results for the Conceptual Understanding Test Instrument

Validation Result	Number of Items
Initial items developed	60
Items aligned with learning objectives and indicators	52
Items requiring revision	5
Items removed	3
Final items used in the study	50
Pretest items	25
Posttest items	25

Initially, 60 test items were developed, comprising 30 pretest items and 30 posttest items. The additional five items for each test were intentionally prepared as reserve items to accommodate possible revisions or removals during the expert validation process. Following expert validation, three items were removed because they did not align with the learning indicators, while five items were revised based on the validator's suggestions. After the revision process, 50 validated items were retained for implementation, consisting of 25 pretest items and 25 posttest items, in accordance with the research design. The remaining validated items were not administered because the study required only 25 items for each test while maintaining representative coverage of the learning objectives.

In addition to expert validation, the reliability of both test forms was examined using Cronbach's Alpha. The pretest instrument obtained a coefficient of 0.870, whereas the posttest instrument obtained 0.844. These values indicate very high internal consistency, confirming that both instruments were reliable for measuring students' conceptual understanding.

5. Media Validation

Prior to implementation, OOPify was evaluated by a programming education expert to determine its suitability for classroom use. The evaluation was conducted using the General Programming Rubric, which assesses four main aspects: solution quality, program design, user interface, and code readability. The results of the media validation are presented in Table 7.

Table 8. Expert Validation Results for the OOPify Media

Criterion	Maximum Score	Obtained Score	Percentage
Solution	10	9	90%
Program Design	10	8	80%
User Interface	10	7	70%
Code Readability	10	9	90%
Average			82.5%

As shown in Table 8, OOPify obtained an average score of 82.5% based on a single-expert appraisal, which falls within the 'very feasible' (81–100%) category [28]. Because this evaluation involved only one expert, the result should be interpreted as formative feedback rather than broad evidence of content or media validity. The highest scores were obtained for the solution and code readability aspects, both receiving 90% ('very feasible'), followed by program design (80%) and user interface (70%), both falling within the 'feasible' (61–80%) category. The expert evaluation indicated that the overall functionality and design of OOPify were appropriate, although several user interface components were identified as requiring further refinement. Overall, this single-expert appraisal provides preliminary evidence supporting the feasibility of OOPify for use in Object-Oriented

Programming learning activities, which should be corroborated by additional experts in future studies.

6. Student Responses toward OOPify

Student responses toward OOPify were collected using a usability questionnaire developed based on Nielsen's five usability attributes [27], covering learnability, efficiency, memorability, errors, and satisfaction. The questionnaire consisted of 10 statements assessed using a five-point Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). A total of 34 students participated in the evaluation. Therefore, the maximum possible score for each dimension was 340 points, while the overall maximum score was 1700 points. The percentage score for each dimension was calculated by comparing the obtained score with the maximum possible score. The results of the student response questionnaire are presented in Table 9.

Table 9. Results of Student Responses toward OOPify

Dimension	Score	Percentage
Learnability	256	75.29%
Efficiency	275	80.88%
Memorability	255	75.00%
Errors	245	72.06%
Satisfaction	278	81.76%
Overall	1309	77.00%

Based on Table 9, the overall student response toward OOPify reached 77.00%, which falls within the 'strong' (61–80%) category based on the interpretation criteria adapted from Riduwan [29], indicating a generally positive perception of its use during practical learning activities. Among the evaluated dimensions, satisfaction (81.76%) and efficiency (80.88%) obtained the highest percentages, both falling within the 'very strong' (81–100%) category, indicating that students were highly satisfied with the media and perceived it as easy to use during practical learning activities. The learnability, memorability, and errors dimensions fell within the 'strong' (61–80%) category, with percentages of 75.29%, 75.00%, and 72.06%, respectively, suggesting that students could learn and remember how to use the media without significant difficulties, although some minor difficulties in error handling were still reported.

The positive student responses indicate that OOPify was perceived as easy to learn and use during practical activities. High scores in the learnability and satisfaction dimensions suggest that students perceived the visual block-based interface as helping them focus on OOP concepts rather than programming syntax. These findings are consistent with previous studies reporting that block-based programming environments can improve learner engagement, usability, and programming confidence, particularly among novice programmers [11], [16], [32], [33]. Furthermore, these findings suggest that OOPify has the potential to serve as a practical learning medium for Object-Oriented Programming in vocational education.

The results of this study indicate that the group-level psychomotor observation scores were higher in the third meeting than in the second meeting, with the average score increasing from 76.18% to 94.26%. These observations indicate that, during the implementation of OOPify, students actively participated in practical learning activities involving block construction, program testing, discussion, and case-study problem solving. These activities provided opportunities for students to demonstrate the moving, manipulating, communicating, and creating dimensions assessed in this study. Previous studies have reported that structured practical learning activities can support the development of psychomotor skills and student engagement [9], [34]. In addition, technology-enhanced learning environments have been recognized as suitable contexts for observing and assessing psychomotor performance during practical learning activities [10].

In the context of OOP education, these findings further support previous studies indicating that one of the major challenges in learning OOP is students' difficulty in connecting abstract concepts with program implementation [1], [2], [4], [35]. Various visual approaches have therefore been developed to help learners establish meaningful relationships between conceptual knowledge and program

implementation [36]. Through block manipulation activities in OOPify, students were able to transform abstract concepts into visual representations that were easier to understand and apply in practice.

Higher group-level psychomotor observation scores were particularly evident in the manipulating and creating dimensions. These observations indicate that the student groups demonstrated greater proficiency in constructing object-oriented program structures, connecting program components, and developing solutions for the given case studies during the observed practical activities. These findings suggest that visual block programming may provide learning experiences that encourage both conceptual engagement with Object-Oriented Programming and practical problem-solving activities. This interpretation is consistent with previous studies reporting that block-based programming environments are associated with higher student engagement, improved conceptual understanding, and the development of programming competencies and practical skills through more intuitive visual representations than conventional programming approaches [32], [37], [38], [39].

Similar findings were reported by Kim et al. [33], who found that block-based Python programming environments help students focus more on programming logic than on syntax, thereby improving learning effectiveness. Other studies have also shown that Scratch-based learning environments can improve problem-solving abilities through visual programming activities that encourage active student participation [40], [41]. In addition, Espinal et al. [42] reported that block-based programming facilitates the transition to text-based programming because learners first develop an understanding of program structures through visual representations.

The findings of this study are consistent with previous research suggesting that OOP learning environments should not only visualize concepts but also integrate practical activities that encourage program implementation skills [7], [21]. In this study, the observed group-level practical performance and the improvement in students' OOP conceptual understanding scores were consistent with this perspective. These findings suggest that OOPify may provide a learning environment that supports connections between conceptual understanding and practical programming activities through structured visual block-programming tasks.

This study has several limitations. First, OOPify was implemented in a single class with a relatively small sample from one vocational high school. Therefore, the findings cannot yet be generalized to broader educational contexts. Second, group-level psychomotor performance was assessed through classroom observations during practical learning activities, making the results dependent on observer judgment. Although the two observers discussed the observation indicators and assessment procedures before data collection to improve scoring consistency, formal inter-rater reliability analysis was not conducted. Consequently, some degree of observer subjectivity may have influenced the psychomotor observation results. Third, because this study employed a one-group pretest–posttest design without a comparison group, the observed changes in students' OOP conceptual understanding scores cannot be attributed solely to the implementation of OOPify. Other factors, such as teacher guidance, collaborative learning activities, repeated testing, or maturation, may also have contributed to the observed outcomes. Fourth, this study relied on a single expert for both content and media validation. While the validator possessed relevant dual expertise in OOP and instructional media, validation by a single expert provides limited evidence of content and media validity compared to validation involving multiple independent experts. Future studies should involve at least two or three experts per instrument to strengthen the validity evidence through inter-expert agreement (e.g., Content Validity Ratio or Aiken's V). Fifth, this study did not obtain formal review from a research ethics committee, and written parental or guardian consent was not collected for participating minors, relying instead on institutional permission and student verbal assent. While the study involved only routine classroom instructional activities and posed minimal risk to participants, the absence of formal ethics committee review and documented parental consent represents an important limitation. Future research involving minors should prioritize obtaining institutional ethics committee approval and documented parental or guardian consent prior to data collection.

Furthermore, the moving dimension reached the maximum score during the final observation, suggesting that this indicator may have been less sensitive in differentiating group-level psychomotor performance. This may be attributed to the relatively basic nature of the observed activities, which were already familiar to vocational high school students. Additionally, because several indicators within the observational rubric (e.g., asking questions, discussing tasks) also capture engagement and collaboration behaviors rather than purely psychomotor competencies, the overall rubric should be interpreted as a broader measure of group-level practical performance. Future studies should therefore refine the psychomotor indicators, particularly within the moving dimension, to better represent practical programming competence and provide greater discrimination among different levels of group performance. Future research should also involve larger and more diverse samples, employ controlled experimental designs, and incorporate learning analytics capable of automatically recording students' interactions while using OOPify. Such developments may provide a more comprehensive understanding of the relationship between visual block-programming activities, group-level psychomotor performance, and students' Object-Oriented Programming skills.

CONCLUSION

This study developed OOPify as a block-based programming multimedia learning tool for Object-Oriented Programming (OOP) using the ADDIE development model. An expert appraisal indicated that the developed multimedia was suitable for classroom implementation, although this conclusion is based on a single expert and should be corroborated by broader validation in future studies. Based on the N-Gain analysis, students' OOP conceptual understanding scores were classified in the moderate category. In addition, the observed group-level practical performance during practical learning activities reached an average score of 85.22%, categorized as very high; however, this finding represents preliminary observational evidence, as no formal inter-rater reliability analysis was conducted. Students also responded positively to OOPify, particularly in terms of learnability, efficiency, memorability, errors, and satisfaction. These findings suggest that OOPify may serve as a practical learning medium for Object-Oriented Programming in vocational high schools. However, because this study employed a one-group pretest–posttest design without a comparison group, the observed changes in students' test scores cannot be attributed solely to the implementation of OOPify. Further research involving larger and more diverse samples and controlled experimental designs is needed to better understand the contribution of OOPify to Object-Oriented Programming learning.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to SMK Negeri 4 Padalarang for facilitating the implementation of this research. The authors also thank the participating teachers and students for their cooperation and valuable contributions throughout the study. Special appreciation is extended to the supervisors for their guidance and constructive feedback during the research process.

AUTHOR CONTRIBUTION STATEMENT

NO: Conceptualization, methodology, software development, investigation, data collection, formal analysis, visualization, and writing-original draft preparation. YP: Investigation, data collection, software development support, and writing-review and editing. MV: Methodology support, formal analysis, visualization, and writing-review and editing. LR: Supervision, validation, methodology review, and writing-review and editing. All authors have read and approved the final version of the manuscript.

AI DISCLOSURE STATEMENT

The author used ChatGPT (OpenAI) during the preparation of this work for language editing and translation purposes. After using the tool, the author thoroughly reviewed and edited the content as needed and takes full responsibility for the content of the publication.

REFERENCES

- [1] Q. A. Batiha, N. A. Abd Majid, N. Sahari, and N. Mohd Ali, "Analysis of the learning object-oriented programming factors," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 5, pp. 5599–5606, 2023, doi: 10.11591/ijece.v13i5.pp5599-5606.
- [2] N. Khudhur, Nurmaya, Y. Hayashi, and T. Hirashima, "Conceptual representation of the source-code to support the learning of object-oriented programming concepts," *International Journal of Information and Education Technology*, vol. 13, no. 12, pp. 1858–1867, 2023, doi: 10.18178/ijiet.2023.13.12.1999.
- [3] Efan, Krismadinata, J. Jama, and R. Mulya, "A systematic literature review of teaching and learning on object-oriented programming course," *International Journal of Information and Education Technology*, vol. 13, no. 2, pp. 302–312, 2023, doi: 10.18178/ijiet.2023.13.2.1808.
- [4] N. Ragonis and R. Shmallo, "The Application of Higher-Order Cognitive Thinking Skills to Promote Students' Understanding of the Use of static in Object-Oriented Programming," *Informatics in Education*, vol. 21, no. 2, pp. 331–352, 2022, doi: 10.15388/infedu.2022.10.
- [5] A. Heinonen, B. Lehtelä, A. Hellas, and F. Fagerholm, "Synthesizing research on programmers' mental models of programs, tasks and concepts: A systematic literature review," *Information and Software Technology*, vol. 164, p. 107300, 2023, doi: 10.1016/j.infsof.2023.107300.
- [6] K. H. Wai et al., "Code Writing Problems for Basic Object-Oriented Programming Study in Java Programming Learning Assistant System," in *Proc. IEEE 12th Global Conference on Consumer Electronics (GCCE)*, Nara, Japan, 2023, pp. 5–6, doi: 10.1109/GCCE59613.2023.10315469.
- [7] A. Tsichouridis, S. Xinogalos, and A. Ampatzoglou, "Educational programming environments for enhancing conceptual design in the object-oriented paradigm: A systematic mapping study," *Journal of Educational Computing Research*, vol. 62, no. 1, pp. 290–322, 2024, doi: 10.1177/07356331231203251.
- [8] A. A. Jamal, A. N. M. Rose, M. Mohamad, A. H. Zakaria, A. Rozaimée, M. K. Awang, and A. R. Mamat, "Psychomotor assessment rubric for computer science courses: Preliminary observation," in *Proc. Int. Conf. Teaching and Learning in Higher Education*, Terengganu, Malaysia, 2017, pp. 133–143.
- [9] F. J. Borrego-Balsalobre, A. Martínez-Moreno, V. Morales-Baños, and A. Díaz-Suárez, "Influence of the Psychomotor Profile in the Improvement of Learning in Early Childhood Education," *International Journal of Environmental Research and Public Health*, vol. 18, no. 23, p. 12655, 2021, doi: 10.3390/ijerph182312655.
- [10] A. Barenji and B. Montreuil, "Intelligent tutoring platform for improving psychomotor skills in STEM workforce education: XR enhanced ITP," *Interactive Learning Environments*, vol. 34, no. 5, pp. 2826–2845, 2026, doi: 10.1080/10494820.2025.2554998.
- [11] Y. Hu, C.-H. Chen, and C.-Y. Su, "Exploring the Effectiveness and Moderators of Block-Based Visual Programming on Student Learning: A Meta-Analysis," *Journal of Educational Computing Research*, vol. 58, no. 8, pp. 1467–1493, 2021, doi: 10.1177/0735633120945935.
- [12] J. A. Rodríguez-Martínez, J. A. González-Calero, and J. M. Sáez-López, "Computational thinking and mathematics using Scratch: An experiment with sixth-grade students," *Interactive Learning Environments*, vol. 28, no. 3, pp. 316–327, 2020, doi: 10.1080/10494820.2019.1612448.
- [13] W.-C. Hsu and J. Gainsburg, "Hybrid and non-hybrid block-based programming languages in an introductory college computer science course," *Journal of Educational Computing Research*, vol. 59, no. 5, pp. 817–843, 2021, doi: 10.1177/073563312098510.
- [14] Ş. Saygıner, "The effects of block-based visual and text-based programming training on students' achievement, logical thinking skills, and motivation," *Journal of Computer Assisted*

- Learning, vol. 39, no. 2, 2023, doi: 10.1111/jcal.12771.
- [15] Q. Yu, K. Yu, and B. Li, "Effects of Block-Based Visual Programming on K–12 Students' Learning Outcomes," *Journal of Educational Computing Research*, vol. 63, no. 1, pp. 64–98, 2025.
 - [16] Y. Lin and D. Weintrop, "The landscape of block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming," *Journal of Computer Languages*, vol. 67, p. 101075, 2021, doi: 10.1016/j.cola.2021.101075.
 - [17] X. J. L. Chin, C. K. Seow, Y. Cai, Y. Zhu, M. Wang, and Q. Cao, "Enhancing student learning and engagement with object-oriented block-based programming tool," *International Journal of Information and Education Technology*, vol. 14, no. 7, pp. 910–919, 2024.
 - [18] K. M. V. Gumonan, C. L. S. Romana, and C. Pantaleon, "Object-Oriented Programming Teaching Methods and Instructional Strategy in Selected Higher Education Institutions in the Philippines," in *Proc. 8th International Conference on Digital Technology in Education (ICDTE)*, 2024, pp. 212–220, doi: 10.1145/3696230.3696255.
 - [19] S. Speth, "Teaching Object-Oriented Programming with the Objects-first Approach – An Experience Report," *SEUH 2023*, doi: 10.18420/seuh2023_04.
 - [20] Y. Qu, F. Li, X. Dou, C. Feng, and B. Li, "Research on Object-Oriented Programming Teaching Evaluation Method Based on Few-Shot Prompt Learning," *Journal of Internet Technology*, vol. 26, no. 7, pp. 893–900, 2025.
 - [21] L. E. Gutiérrez, C. A. Guerrero, and H. A. López-Ospina, "Ranking of problems and solutions in the teaching and learning of object-oriented programming," *Education and Information Technologies*, vol. 27, pp. 7205–7239, 2022, doi: 10.1007/s10639-022-10929-5.
 - [22] S. López-Pernas and M. Saqr, "Bringing Synchrony and Clarity to Complex Multi-Channel Data: A Learning Analytics Study in Programming Education," *IEEE Access*, vol. 9, pp. 159422–159436, 2021, doi: 10.1109/ACCESS.2021.3134844.
 - [23] K. S. Huda, L. A. Alfitri, R. Hamzah, and L. S. Riza, "Learning Evaluation Using Block Programming on Object-Oriented Programming Materials to Improve Cognitive Skills," *Indonesian Journal of E-Learning and Multimedia (IJOEM)*, vol. 4, no. 3, pp. 223–236, 2025, doi: 10.58723/ijoem.v4i3.473.
 - [24] S. Arikunto, *Dasar-Dasar Evaluasi Pendidikan*, 3rd ed. Jakarta, Indonesia: Bumi Aksara, 2021.
 - [25] R. L. Ebel and D. A. Frisbie, *Essentials of Educational Measurement*, 5th ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.
 - [26] L. W. Trowbridge, R. W. Bybee, and R. B. Sund, *Becoming a Secondary School Science Teacher*, 3rd ed. Columbus, OH, USA: Charles E. Merrill Publishing Company, 1981.
 - [27] J. Nielsen, *Usability Engineering*. Boston, MA, USA: Academic Press, 1993.
 - [28] Riduwan, *Skala Pengukuran Variabel-Variabel Penelitian*. Bandung, Indonesia: Alfabeta, 2023.
 - [29] Riduwan, *Skala Pengukuran Variabel-Variabel Penelitian*. Bandung, Indonesia: Alfabeta, 2012.
 - [30] R. R. Hake, "Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses," *American Journal of Physics*, vol. 66, no. 1, pp. 64–74, Jan. 1998, doi: 10.1119/1.18809
 - [31] F. Bailie, B. Marion, and D. Whitfield, "How rubrics that measure outcomes can complete the assessment loop," *Journal of Computing Sciences in Colleges*, vol. 25, no. 6, pp. 15–25, 2010.
 - [32] D. Sun, C. K. Looi, Y. Li, C. Zhu, C. Zhu, and M. Cheng, "Block-based versus text-based programming: A comparison of learners' programming behaviors, computational thinking skills and attitudes toward programming," *Educational Technology Research and Development*, vol. 72, pp. 1067–1089, 2024, doi: 10.1007/s11423-023-10328-8.

- [33] Y. Kim, J. Kim, and W. Lee, "Effect of Block-Based Python Programming Environment on Programming Learning," *Applied Sciences*, vol. 13, no. 19, p. 10898, 2023, doi: 10.3390/app131910898.
- [34] D. Roşu, F. Cojanu, P.-F. Vişan, N. Samarescu, M. A. Ene, R.-I. Muntean, and V. E. Ursu, "Structured Program for Developing the Psychomotor Skills of Institutionalized Children with Special Educational Needs," *Children*, vol. 11, no. 1, p. 102, 2024, doi: 10.3390/children11010102.
- [35] T. Diaz-Leyva and O. Chamorro-Atalaya, "Analysis of learning difficulties in object oriented programming in systems engineering students at UNTELS," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 6, pp. 1704–1709, 2020, doi: 10.25046/aj0506203.
- [36] V. Lian, E. Varoy, and N. Giacaman, "Learning object-oriented programming concepts through visual analogies," *IEEE Transactions on Learning Technologies*, vol. 15, no. 1, pp. 78–92, 2022, doi: 10.1109/TLT.2022.3154805.
- [37] J. Cárdenas-Cobo, C. Vidal-Silva, and N. Máquez, "Dataset on programming competencies development using Scratch and a recommender system in a non-WEIRD primary school context," *Data*, vol. 10, no. 6, p. 86, 2025, doi: 10.3390/data10060086.
- [38] H. Suriyaarachchi, P. Denny, and S. Nanayakkara, "Scratch and Sense: Using real-time sensor data to motivate students learning Scratch," in *Proc. 53rd ACM Technical Symposium on Computer Science Education (SIGCSE 2022)*, 2022, pp. 983–989, doi: 10.1145/3478431.3499316.
- [39] T. Talan, Y. Doğan, Y. Kalinkara, and V. Batdi, "To block or not to block? A meta-analysis of effectiveness of block-based programming," *Research in Science & Technological Education*, pp. 1–20, 2025, doi: 10.1080/02635143.2025.2593936.
- [40] O. Erol and N. S. Çırak, "The effect of a programming tool Scratch on the problem-solving skills of middle school students," *Education and Information Technologies*, vol. 27, pp. 4065–4086, 2022, doi: 10.1007/s10639-021-10776-w.
- [41] J. Cárdenas-Cobo, A. Puris, P. Novoa-Hernández, Á. Parra-Jiménez, J. Moreno-León, and D. Benavides, "Using Scratch to Improve Learning Programming in College Students: A Positive Experience from a Non-WEIRD Country," *Electronics*, vol. 10, no. 10, p. 1180, 2021, doi: 10.3390/electronics10101180.
- [42] A. Espinal, C. Vieira, and V. Guerrero-Bequis, "Student ability and difficulties with transfer from a block-based programming language into other programming languages: a case study in Colombia," *Computer Science Education*, vol. 33, no. 4, pp. 567–599, 2023, doi: 10.1080/08993408.2022.2079867.