2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #53029

Exploring Peer-to-Peer Evaluation with an AI-Supported Learning Tool in an Introductory Programming Course: An MSI Case Study

Edward Dillon, University of Maryland Baltimore County

Edward Dillon is an Associate Professor in the Department of Information System at the University of Maryland, Baltimore County (UMBC). Edward received his B.A. in Computer and Informational Science from the University of Mississippi in 2007. He would go on to obtain his Masters and Ph.D. in Computer Science from the University of Alabama in 2009 and 2012, respectively. Prior to his arrival to UMBC in August 2024, Dr. Dillon served as a Computer Science Instructor at Jackson State University (2012-2013), and a Postdoctoral Researcher at Clemson University (2013-2014) and the University of Florida (2014-2016), and Assistant/Associate Professor in the Department of Computer Science at Morgan State University (2017-2024). His research focuses on human-centered computing with emphasis on computing education, broadening participation in computing, and tech workforce prep.

Dr. Patricia Ordóñez, University of Maryland Baltimore County

Patricia Ordonez is an Associate Professor in Information Systems at the University of Maryland, Baltimore County since 2022. She received her BA in Hispanic and Italian Studies from Johns Hopkins University. She received her MS and PhD in Computer Science.

Carine Marette, Indiana University-Bloomington

Carine Marette is a doctoral candidate in Instructional System Technology at Indiana University Bloomington's School of Education, specializing in AI-enhanced engineering pedagogy and learning analytics. Her research examines how constrained artificial intelligence environments can function as measurement affordances for observing transfer-relevant behaviors in authentic learning contexts. She developed the Vector Framework for multi-source learning telemetry, integrating behavioral data from AI interaction logs and peer assessment platforms to measure transfer readiness, productive struggle, and metacognitive development. Her work bridges engineering education, cognitive load theory, and educational measurement, with particular focus on developing validity arguments for AI-mediated assessment in project-based learning environments. Carine co-created with Kritik360 and VisibleAI platforms, which provide transparency-oriented authorship tracking and structured peer evaluation systems. Her dissertation investigates adaptive constraint design for differentiated AI scaffolding across learner distributions in graduate engineering courses.

Ben Cohen, University of Maryland Baltimore County
Omobolanle Niyi Owoeye, University of Maryland Baltimore County
Kevin Lemus, University of Maryland Baltimore County

Kevin Lemus is a graduate assistant researcher and Master's student at the University of Maryland, Baltimore County (UMBC). Kevin earned his Bachelor of Science in Information Systems along with a Certificate in User Experience, Web, and Mobile Development from UMBC in 2024. He is currently advancing his expertise as a candidate for a Master of Science in Human-Centered Computing, with an expected completion date of December 2026.

SRUSHTI RAJESH DHARMALE, University of Maryland Baltimore County

Exploring Peer-to-Peer Evaluation with an AI-Supported Learning Tool in an Introductory Programming Course: An MSI Case Study

Abstract

With the growing prevalence of generative artificial intelligence and its capacity to instantly generate computational coding solutions through large language model-based queries, concerns have emerged among educators regarding the potential attrition of students' critical thinking and analytical problem-solving abilities. Existing education literature indicates that peer-to-peer evaluation can function as a pedagogical intervention to promote deeper learning and improve programming performance, particularly among novice programmers. These interventions provide opportunities for students to engage in delivering and receiving constructive feedback, thereby fostering communication skills, reflective thinking, and reinforcement of conceptual understanding in programming contexts. Even in the age of artificial intelligence, new tools are being developed to facilitate peer-to-peer evaluation and support collaborative learning among students.

To further investigate the impact of peer-to-peer evaluation on novice programmers, a case study was conducted in an introductory programming course at a minority-serving institution in the Mid-Atlantic United States during the Fall 2024 semester. The course integrated Kritik360, a commercial-based and AI-supported peer assessment platform, to examine the influence of structured group discussions implemented as weekly learning activities among Information Systems majors in this introductory programming course.

Kritik360 is designed to foster student engagement and promote the development of critical thinking and evaluative skills as students engage with course content through collaborative learning tasks. The primary objective of this study was to assess the effectiveness of Kritik360 in enabling students to engage in peer-to-peer evaluation within assigned groups as they develop foundational computational thinking skills and understanding core data structures in Java programming. To evaluate the impact of Kritik360 on students' learning experiences, a survey was administered consisting of questions designed to collect both quantitative and qualitative data regarding students' perceptions and interactions with the tool. A total of 31 students completed the survey.

The findings indicated that students held moderately positive perceptions of Kritik360's effectiveness in enhancing their evaluative skills. Additionally, the tool was perceived as beneficial for supporting programming and code comprehension practices. However, when asked about potential areas for improvement, students noted certain aspects of Kritik360's usability and

feature capacity that could be further refined to better support the learning process.

1. Introduction

Learning to program can be challenging, yet it is an essential skill for computing majors to achieve both academic and professional success. Prior literature has emphasized that a novice's ability to develop proficient mental models of programming is a critical factor in learning success [1]. Such mental models encompass the capacity for effective code comprehension [2], critical thinking [3, 4, 5], and problem-solving [6, 7]. In addition, code evaluation and review are fundamental skills for novice programmers, as they support the development of both concrete (functional) understanding and abstract knowledge that underpin deeper code comprehension [8, 9].

With the emergence of artificial intelligence (AI), designing learning environments that effectively foster novice learning and programming skill development presents a new set of challenges, particularly with respect to academic honesty, integrity, and ethical considerations [10, 11]. Nevertheless, the presence of AI also offers opportunities to enhance computational learning for novices when used appropriately and with intentional pedagogical guidance.

This article presents a case study that integrated Kritik360, a commercial-based AI-supported peer-to-peer evaluation tool, into two sections of a 17-week introductory Java course at a minority-serving institution in the Mid-Atlantic United States during the Fall 2024 semester. The objective of the study was to examine Kritik360's capacity to support student learning and programming skill development through structured peer-to-peer interactions. To assess its impact, a survey was administered near the end of the semester to capture students' perceptions, perspectives, and overall views regarding Kritik360's effectiveness in supporting programming pedagogy practices involving code review and code comprehension.

2. Literature Review

2.1 Integrating AI Tools to Support Student Learning

AI tools are increasingly being integrated into higher education to support student learning through personalization, feedback, and instructional decision-making. In a systematic review of empirical research on AI in online higher education, Ouyang et al. [12] found that AI applications are predominantly used to predict students' learning status or performance, recommend learning resources, automate assessment, and enhance learning experiences. Notably, prediction and recommendation systems accounted for the majority of applications reviewed, indicating that AI is most often deployed as a supportive layer that monitors learning data and provides adaptive guidance rather than as a substitute for core learning activities [12]. This pattern suggests that, in current practice, AI primarily functions as an augmentation tool for instruction and learning rather than as a mechanism for replacing human-centered pedagogical processes.

More recent synthesis work situates AI integration within established learning theory and emphasizes that its educational impact depends on instructional design rather than technology alone. Using Vygotsky's Zone of Proximal Development (ZPD) as an organizing framework, Cai

et al. [13] conducted a systematic review of 158 empirical studies and concluded that AI tools can support learning by enabling scaffolding through social and technological interaction, communication, and structured instructional support. Their review further reports consistent associations between AI-supported learning environments and improvements in student motivation, engagement, learning support, and academic performance, while also highlighting the importance of educator professional development and ethical deployment of AI in instructional contexts.

At the implementation level, broad work on artificial intelligence in education emphasizes both substantial benefits and significant challenges. For example, Madanchian et al. [14] argue that AI technologies can enhance learning outcomes through personalized learning experiences, increased engagement, and improved instructional efficiency. At the same time, their review highlights critical institutional and ethical considerations, including data privacy and security, algorithmic bias and fairness, accessibility and inclusivity, and the need for sustained professional development for educators. Together, these findings suggest that the educational value of AI depends not only on technological capability, but also on responsible, well-supported, and pedagogically aligned implementation.

2.2 Integrating AI Tools to Support Computational Learning

In programming education, generative AI and code-assistance tools introduce both significant opportunities for learning support and substantial risks of overreliance. In a systematic literature review of 33 empirical studies, Dwiantoro et al. [15] conclude that AI tools can enhance creativity, collaboration, and student engagement through personalized feedback and adaptive learning environments, while simultaneously raising concerns about diminished critical thinking and ethical issues such as bias and authorship. Importantly, the review emphasizes that effective implementation depends on pedagogical strategies that promote reflection and responsible tool use, rather than substituting AI for core learning processes.

Amoozadeh et al. [16] looked at levels of trust in generative AI tools among computer science students at universities in India and the United States. They found that 47 percent of participants reported trusting generative AI tools using the trust framework put forth by Körber [17], while 16 percent stated they did not trust the tools. Both Becker et al. [18] and Lau and Guo [19] examined these tools from a professor's perspective, with each study focusing on changes that could be made, or were being made, to adapt to the sudden availability of generative AI tools. The former argued that instruction should shift toward emphasizing reading and analyzing, rather than writing, code, suggesting that such a shift would validate the pedagogical approaches suggested by Xie et al. [20] prior to the launch of ChatGPT. Becker et al. [18] also argue that such tools will force educators to confront the ethical issues surrounding the use of such tools and discuss these ethical issues with their students. Lau and Guo surveyed 20 introductory programming instructors across nine different countries. This revealed that several instructors indicated that they felt formally integrating AI tools into their computer science curricula would help distinguish their respective institutions, that the ability of students to produce code using AI tools would allow for more advanced topics to be covered in CS 1 and CS 2 courses, and that they were planning to shift their assignments and assessments away from writing code and toward analyzing and critiquing provided code [19].

Literature has also shown that generative AI in conjuntion with project based learning interventions can be used to assist with code generation, reasoning, and task guidance, helping learners manage cognitive load during program development while remaining actively engaged in computational thinking processes. Gou and Ren [21], for instance, examined AI usage through project based learning interventions to assess whether it supports students' ability to decompose problems, design algorithms, and reflect on solutions.

Related work also highlights the potential of generative AI to support computational thinking tasks through explanation and feedback. Benedetti [22] argues that generative AI can provide timely support for problem solving activities, such as Bebras style tasks, by assisting learners in planning and reasoning about solutions rather than simply producing final answers. At the same time, this work emphasizes the importance of careful instructional design, noting that unstructured use of generative AI may lead to overreliance and reduced engagement with underlying reasoning processes. Other interventions extend beyond problem solving support to examine conversational and representational AI tools for computational learning. Suh and An [23] present a generative conversational AI system, CodeToon, which enables learners to transform code into narratives and comics. By mapping programming constructs to natural language stories and visual representations, this approach supports learners in moving across levels of abstraction, a central component of computational thinking, while emphasizing sensemaking over correctness.

Existing literature has also investigated the direct impact of widely used generative AI tools, such as ChatGPT, on computational learning. In an undergraduate programming course, students reported using ChatGPT primarily to understand programming concepts, learn syntax, and generate algorithmic ideas rather than to directly complete assignments [24]. While students generally perceived ChatGPT as helpful, concerns were raised regarding incorrect outputs and the potential impact on independent skill development, suggesting that unguided use may undermine learning. Yilmaz and Yilmaz [25] found that students who used ChatGPT during programming instruction demonstrated higher computational thinking skills, programming self-efficacy, and motivation compared to a control group. In this study, ChatGPT, based on the GPT-3.5 architecture, functioned as an individualized learning support tool that provided explanations, feedback, and coding assistance during hands on practice.

Conceptual and integrative perspectives have also reinforced the alignment between artificial intelligence and computational thinking. Tian [26] proposes that computational thinking and artificial intelligence rely on shared cognitive processes, including abstraction, decomposition, and debugging, providing a theoretical rationale for their integration in educational contexts. Similarly, a systematic review by Weng et al. [27] synthesizes empirical studies integrating AI and computational thinking, identifying tools such as intelligent tutoring systems, chatbots, automated feedback systems, and generative AI models. Importantly, this review highlights that learning outcomes are shaped less by the presence of AI tools themselves and more by the instructional designs through which those tools are embedded.

Building on these perspectives seen in literature, the authors argue that a practical instructional response is to shift learning activities toward tasks that require explanation, evaluation, and revision, forms of work that are less amenable to direct "outsourcing" to AI and more supportive of durable conceptual understanding. In this way, ethical and instructional considerations can be leveraged to strengthen students' critical evaluation of AI-generated outputs and their

professional reasoning in programming contexts.

2.3 Peer-to-Peer Learning

Peer-to-peer learning approaches, particularly peer assessment, have been widely studied as mechanisms for improving feedback processes, self-regulation, and learning outcomes. A large-scale meta-analysis of 54 experimental and quasi-experimental studies found that peer assessment has a statistically significant positive effect on academic performance, with a small-to-moderate overall effect size ($g = 0.31$), compared to both no assessment and teacher-only assessment [28]. Beyond achievement effects, peer assessment is theorized to support learning by engaging students in evaluative judgment, comparison with peer work, and reflection on quality criteria. Importantly, Double et al. [28] also emphasize the role of peer assessment as a scalable formative assessment strategy, particularly in instructional contexts where instructor feedback is constrained by class size, workload, or resource limitations.

However, the effectiveness of peer assessment depends critically on a range of human and social factors that shape students' engagement, perceptions of fairness, and the quality of feedback processes. In a comprehensive review, Panadero et al. [29] synthesize evidence on both intrapersonal factors, including motivation, self-efficacy, emotions, comfort, and fairness perceptions, and interpersonal factors, including trust in peers, psychological safety, social connections, and interdependence, all of which influence how peer assessment is enacted and experienced. Because peer assessment is inherently a social activity, students' perceptions of credibility, trust, and safety play a central role in determining how seriously they engage with reviewing, interpreting, and revising work.

In introductory programming courses, peer assessment may be especially valuable because novice learners benefit from viewing diverse solution strategies and from articulating standards of what constitutes high-quality code. Moreover, the act of providing feedback requires students to externalize evaluation criteria and justify their reasoning, processes that support the development of metacognitive awareness and evaluative judgment [29].

Extending this focus on learner perceptions of peer assessment, Matejić [30] reports findings from a survey-based study conducted after a peer-assessment phase in an undergraduate web programming course. The results indicate that students generally perceived both giving and receiving peer feedback as beneficial for improving project quality and supporting personal development, particularly in terms of evaluation and self-evaluation skills and reflective thinking. Students also reported perceived gains in critical thinking, self-confidence, and the quality of academic communication, suggesting that peer assessment can contribute not only to technical refinement of work but also to broader cognitive and communicative development.

Reliability and feasibility are persistent concerns in peer-based evaluation, particularly when comparing peer and self assessments to expert judgments. In an engineering education context, Power and Tanner [31] examined peer assessment and self-assessment exercises relative to expert assessor scores and found that peer assessment demonstrated relatively high reliability compared with expert evaluations, whereas self-assessment exhibited lower alignment with expert scores. They also reported that the quality of peer feedback was strongly associated with student performance, underscoring that evaluative quality is what really matters for learning outcomes.

Despite indications of peer assessment's potential, questions of feasibility and reliability of peer-generated scores and feedback remain central design considerations for effective implementation, especially in contexts where expert validation is limited but consistent and constructive peer engagement is desired.

2.4 Code Reviews

Peer code review (PCR) represents a discipline-authentic form of peer assessment that mirrors professional software development practices, where code is routinely reviewed by peers to ensure quality before deployment. In a systematic literature review of recent PCR implementations in education, Ismail et al. [32] categorized existing work into three major strands: empirical studies, approach-focused studies, and system-based implementations. Across these strands, the review identifies student participation and engagement, review quality, and programming skill development as the three central outcome dimensions investigated in the literature. Importantly, the authors emphasize that the success of PCR in educational contexts depends heavily on student engagement, noting that low participation can lead to poor review quality and undermine the instructional purpose of the activity. This synthesis suggests that effective PCR designs must attend not only to technical infrastructure or review procedures, but also to motivational and participation structures that sustain meaningful student involvement.

Tool support can help address several structural barriers to peer review in programming courses, particularly in in-class settings where students require rapid feedback and instructors cannot feasibly review all student work. Wang et al. [33] introduced PuzzleMe, a web-based platform designed to support in-class programming exercises through two peer assessment mechanisms: live peer testing and live peer code review. The system enables students to create, verify, and share test cases, and uses intelligent grouping based on solution correctness and diversity to promote meaningful code discussions. In a multi-stage classroom deployment, students reported that these peer assessment activities helped them identify defects in their code, correct conceptual misunderstandings, and learn alternative solution strategies from their peers, with usage data showing improved problem completion following peer review interactions [33].

In introductory programming courses, peer assessment can be challenging because novice students may initially lack sufficient domain knowledge to evaluate their peers' work reliably. Al-Khalifa and Devlin [34] explicitly address this concern in their empirical study of peer assessment with novice programmers, showing that structured supports such as rubrics and marking schemes can scaffold the assessment process and make peer assessment feasible in early-stage programming courses. Importantly, their results indicate that participation in peer assessment activities had a positive effect on students' subsequent achievement, as measured by performance on a later course exam. Complementing these findings, Matejić [30] reports in the context of a web programming project that students perceived peer feedback as supporting not only technical improvement, but also reflective engagement and the development of professional and collaborative skills. Brown et al. [35] and Hundhausen et al. [8], respectively, highlight additional skills that students develop through the integration of peer coding reviews into early computing courses, including communication, teamwork, peer-to-peer engagement, and interpersonal skills. Altogether, these studies suggest that while peer assessment with novice learners requires careful scaffolding, it can yield cognitive, metacognitive, and soft skill benefits

when thoughtfully designed.

2.5 AI-Supported Peer Assessment Tools in Programming Courses

Within the current AI-augmented programming context, peer review and rubric-based evaluation have gained renewed attention as potential responses to AI-enabled plagiarism and the increasing difficulty of validating student authorship using traditional grading practices. Berrezueta-Guzman et al. [36] argue that AI coding assistants fundamentally challenge conventional assessment models and motivate a shift toward approaches that emphasize evaluative judgment rather than mere solution production. In their empirical study of a structured, rubric-based, and anonymized peer review process in a large introductory programming course, they demonstrate that peer assessments can approximate instructor evaluations with moderate accuracy, while also fostering student engagement, reflective comparison, and evaluative thinking. Together, these findings position structured peer assessment not only as a pragmatic response to AI-related integrity concerns, but also as a pedagogically meaningful mechanism for sustaining critical judgment in AI-rich learning environments.

AI-supported peer assessment tools also extend the peer assessment model by embedding features intended to scaffold reviewing, such as structured workflows, rubric guidance, and feedback supports; potentially increasing the consistency and usefulness of peer feedback. In the context of this study, Kritik360 is used to structure a create–evaluate–feedback cycle and support students' engagement with peer critique as part of learning introductory programming.

Evidence on AI-assisted peer assessment platforms such as Kritik™ highlights several promising outcomes for student learning and engagement. In a multi-course, cross-sectional study, students demonstrated stronger social learning behaviors, with the platform fostering active interaction rather than passive assignment submission Stylianides et al. [37] found that students with lower GPAs reported more favorable experiences with Kritik™, particularly in areas such as adapting their learning strategies, planning their study approach, and articulating their understanding—suggesting that peer learning can serve as a valuable academic support system. Female-identifying students consistently rated the platform higher in terms of ease of use, collaboration, and learning outcomes, pointing to the potential of structured peer learning to create more inclusive educational environments. Finally, the findings underscore that technology like Kritik™ does not replace pedagogy but enhances it—scaling effective teaching practices through thoughtful instructional design.

Taken together, prior research supports the rationale for examining AI-supported peer assessment in introductory programming. Peer assessment is associated with improved academic performance [28], but its effectiveness depends critically on human and social conditions such as trust, psychological safety, and students' engagement with the process [29]. In programming education specifically, peer code review has been shown to strengthen learning outcomes, yet its success hinges on sustained participation and meaningful engagement [32]. At the same time, the rise of AI coding copilots intensifies the need for pedagogies that foreground critique, reflection, and evaluative judgment rather than mere solution production [36, 15]. Finally, evidence from AI-supported peer assessment platforms suggests that such tools do not function as one-size-fits-all solutions and that their impact depends heavily on instructional design and

learner context [37, 33], reinforcing the need for carefully structured, pedagogically grounded implementations in novice programming courses.

3. Kritik360 and Visible

Kritik360 is an online peer-assessment platform designed to engage students in structured peer learning while reducing grading workload for instructors. Within a Kritik activity, students complete three stages (Create, Evaluate, and Feedback) in which they submit an assignment ("creation"), anonymously assess several peers' submissions using an instructor-provided rubric and written comments ("evaluate"), and then rate and respond to the quality of the feedback they received ("feedback"), emphasizing feedback that is both motivational and critical [38, 39]. The platform's design aligns with peer-learning approaches that aim to develop higher-order skills by requiring students to interpret criteria, judge quality, and communicate actionable suggestions.

To support effective peer evaluation, Kritik360 provides instructor-controlled rubrics that can be created manually or generated using the platform's AI rubric feature, which produces rubric criteria based on the assignment description and learning objectives [40]. Instructors can also scaffold peer learning during the evaluation stage by spotlighting exemplary student creations or evaluations, sharing models of quality work and feedback to clarify expectations, without automatically assigning grades [41]. More broadly, Kritik encourages timely, explicit, and constructive feedback practices through the TEACH model as a framework for instructor facilitation [38].

A core element of Kritik's assessment model is its approach to accountability and grading accuracy. Students receive scores associated with their participation across creation, evaluation, and feedback stages [42]. In addition, Kritik uses grading score/grading power concepts to weight peer evaluations, such that feedback from more accurate evaluators carries greater influence on resulting scores [43, 39]. Calibration and ongoing adjustments (e.g., activity-to-activity updates) are used to reflect changes in students' evaluator accuracy over time, allowing instructors to maintain oversight while distributing evaluative work across the class [43, 39].

In this study, Kritik360 was paired with VisibleAI, a transparency-oriented tool used during the creation stage that provides an authorship-focused breakdown of how AI contributed to the submitted work, with the goal of supporting responsible AI use and reducing reliance on detection-only approaches [40, 44]. This emphasis on transparency is consistent with calls in educational AI for explainability that supports appropriate trust, interpretation, and oversight of AI-supported learning tools [45].

Finally, the use of tool-supported peer review in programming contexts is consistent with prior systems research. For example, PuzzleMe demonstrated that structured peer testing and peer code review during in-class programming exercises can support novice learning by helping students detect errors, compare alternative solutions, and build confidence [33]. Lately, Sofjan et al. [46] showed that Kritik360 has strong correlation between the instructor evaluations and weighted peer scores, demonstrating that weighted peer ratings is a collaborative system that enhances reliability. Kritik is a reliable, effective and scalable tool for the instructor as scores can be automatically weighted with a given score by a piecewise linear equation [46]. These parallels

Table 1: Overview of Course Topics that Required Kritik360

Chapter	Topic	Integration of Kritik360
1	Introduction to Computers, Programs, and Java	No
2	Elementary Programming	Yes
3	Selections	Yes
4	Mathematical Functions, Characters, and Strings	Yes
5	Loops	Yes
6	Methods	No
7	Single-Dimensional Arrays	No
8	Multi-Dimensional Arrays	No
9	Objects and Classes	Yes

further motivate the use of a structured peer-assessment workflow such as Kritik360 in introductory programming courses, particularly when the instructional goal is to strengthen students' evaluative judgment and feedback literacy.

4. Methods

This initiative implemented a targeted intervention in two of five introductory Java programming course sessions during the Fall 2024 semester. The objective of this initiative was to examine the impact of Kritik360 on students' mastery of programming concepts covered in the course. The study design employed a mixed-methods approach, utilizing a survey that included both Likert-scale and open-ended questions to collect quantitative measures and qualitative insights from students. For the Likert-scale questions, quantitative analyses were conducted using mean scores to represent students' ratings. For the open-ended responses, qualitative thematic coding [47] was applied to categorize student feedback. To ensure objectivity and reliability, four of the authors independently analyzed the same set of responses for these open-ended questions before engaging in discussions to reach a consensus on the final themes representing the students' feedback.

Kritik360 was integrated into a targeted set of homework assignments aligned with specific course topics. Upon completing each assignment as a Java programming solution, students were required to upload their work to Kritik360. For each student submission, the platform assigned four peers to review, analyze, and critique the solution, providing feedback, suggestions, and recommendations for improvement where necessary. Table 1 outlines the course topics by chapter and identifies the corresponding topics for which students were required to use Kritik360 as part of their homework assignments. Kritik360 was incorporated into only a subset of course topics due to differences in topic complexity and the overall course workload. After careful consideration, the first and second authors concluded that limiting the use of Kritik360 to a selected number of topics was the most effective approach.

4.1 In-Course & Survey Evaluations

The survey was administered toward the end of the semester using SurveyMonkey. Nine specific questions (Q1–Q9) were designed to capture students' experiences and perceptions of Kritik360. From these two sections, a total of 31 students completed this survey.

The initial question (*Q1*) asked students to report their overall attitudes toward peer-to-peer evaluation using a 7-point Likert scale ranging from **Absolutely Not Like** to **Absolutely Like**. The remaining Likert-scale questions (*Q2-Q7*) also employed 7-point scales, ranging from **Absolutely Not** to **Absolutely Yes**. The following Likert-scale questions were asked:

- Q1: How do you feel about peer-to-peer evaluation?
- Q2: Did Kritik improve your critical thinking about programming?
- Q3: Do you think that Kritik has helped you build your critical thinking skills in general?
- Q4: Do you think that Kritik has helped you build your evaluation skills?
- Q5: Do you think that Kritik has helped you build your writing skills?
- Q6: Do you think that Kritik has helped you build your communication skills?
- Q7: Do you think that Kritik has helped you build your programming skills in general?

The objective of these questions was to assess students' perceptions of Kritik360's impact on their computational learning, programming skill development, and related professional skills, as well as to determine whether the tool supported these learning outcomes.

In addition, the survey included two open-ended questions (*Q8* and *Q9*) designed to elicit detailed student reflections on their experiences with Kritik360 and to gather suggestions for improving its use in future introductory programming courses. These two open-ended questions are as follows:

- Q8: Please elaborate on your experience(s) with Kritik in the classroom.
- Q9: If you feel Kritik has potential in the classroom, how do you think it could be improved?

The Result section details the results that were obtained in relation to these nine questions, and highlight how the results addressed each research question.

5. Results & Discussion

Table 2 presents a descriptive summary of responses from the 31 students across all seven Likert-scale questions. For Q1, which examined students' general attitudes toward peer-to-peer evaluation, the mean score was **4.87/7.00**. The mean scores for the remaining questions were **4.84/7.00** for Q2, **4.58/7.00** for Q3, **5.16/7.00** for Q4, **4.42/7.00** for Q5, **4.19/7.00** for Q6, and **4.74/7.00** for Q7, respectively.

To further analyze the mean scores for Q2–Q7, additional statistical analyses were conducted to determine whether certain scores were significantly higher than others. A one-way ANOVA was

Table 2: Mean Scores & Standard Deviations for Q1-Q7

Question	Mean	SD
Q1	4.87	1.52
Q2	4.84	1.73
Q3	4.58	1.73
Q4	5.16	1.37
Q5	4.42	1.78
Q6	4.19	1.76
Q7	4.74	1.75
SD = standard deviation		

Table 3: Theme Classifications for Q8 (using Thematic Analysis)

Theme Classification	N*
Favorable	16
Neutral	4
Unfavorable	6
* *denotes 26 responses in total*	

performed on these six questions, which did not reveal a statistical significance. Two-tailed T-Tests were also conducted to determine any statistical significance when comparing each question in pairs. Only one pair of questions indicated a statistical significance at a 95% confidence interval ($p < 0.05$), Q4 vs. Q6.

These results suggest that students held generally moderate perceptions of peer-to-peer evaluation. Further examination of their views regarding the impact of Kritik360 on various aspects of the learning process (Q2–Q7) similarly revealed predominantly moderate ratings across most attributes. Notably, students reported a slightly higher mean for Q4, which relates to the development of evaluation skills. This difference was also found to be statistically significant when compared to Kritik360's perceived impact on communication skills.

For Q8, thematic analysis was employed to categorize each response as *Favorable*, *Unfavorable*, or *Neutral*. These responses reflect feedback from 26 of the 31 students who answered this question. Table 3 summarizes the themes identified from these responses. These results indicate that most students held a favorable perception of their experience with Kritik360, although a subset of students reported neutral or unfavorable perceptions.

To further examine these responses, a second round of thematic analysis was conducted to identify themes that help elucidate the factors contributing to students' favorable or unfavorable perceptions of their experiences with Kritik360. To focus specifically on these perspectives, neutral responses were omitted from this phase of the analysis. The analyzed responses reflect feedback from 22 of the 31 students whose responses were initially determined to be either favorable or unfavorable. Table 4 summarizes the themes identified from these responses, organized by the favorability of the students' initial perceptions.

Table 4: Detailed Theme Classifications for Q8 (using Thematic Analysis)

	Theme Classification	**N**
Favorable (N=16)	Communication	3
	Comprehension	8
	Observation	1
	Other	2
	Dual Classification	2
Unfavorable (N=6)	Suitability	2
	Usability	4

Favorable responses were categorized into five themes: *Communication*, *Comprehension*, *Observation*, *Other*, and *Dual Classification*. Among these, Comprehension exhibited the highest frequency of responses, followed by Communication. Other and Dual Classification were the next most frequent, while Observation yielded the fewest responses. Within the Dual Classification category, two student responses were jointly coded as Comprehension and Observation and Comprehension and Communication, respectively. Overall, these findings suggest that Kritik360 was perceived as most impactful in supporting students' development of computational comprehension. The quote below provide a notable example of a response classified under the Comprehension theme:

> ***Comprehension:*** *"At first, I struggled a lot with programming and with the Kritik prompts. But being able to receive feedback and how others were able to solve the prompt, I learned a lot."*

Unfavorable responses were categorized into two themes: Suitability and Usability, with Usability exhibiting the higher frequency. The quote below provide a notable example of a response classified under the Usability theme:

> ***Usability:*** *"Kritik is a little annoying to me because the assignments aren't due at one time, you have to go in and do an assignment and then wait to do the next one. I would rather just get them all done at my own pace."*

For Q9, thematic analysis was also conducted to gather themes representing how the students believed Kritik360 could be improved for future usage. The responses reflect feedback from 23 of the 31 students who completed this question. Table 5 summarizes the themes identified from these responses.

These responses were categorized into five themes: *Communication*, *Effectiveness*, *Features*, *Other*, and *No Improvement Needed*. Among these, Features exhibited the highest frequency of responses, followed by Effectiveness. No Improvement Needed was the next most frequent category, followed by Communication. A single response was classified under Other. The quote below provide a notable example of a response classified under the Features theme:

> ***Features:*** *"I think Kritik could be improved by automatically transferring the color-coded text from JGrasp and other coding sites that way you don't have to manually do it. "*

Table 5: Detailed Theme Classifications for Q9 (using Thematic Analysis)

Theme Classification	**N***
Communication	3
Effectiveness	6
Features	9
Other	1
No Improvement Needed	4
** denotes 23 responses in total*	

Overall, the survey results provide meaningful feedback on Kritik360's impact on pedagogical computational learning in the classroom. In this study, Kritik360 demonstrated potential as a reliable tool for fostering code evaluation practices and supporting the development of programming comprehension skills. At the same time, areas for improvement were identified, particularly related to usability and feature limitations with respect to time on task and the platform's adapative capacity to support the transfer and integration of coding content from commonly used coding and text editors.

6. Limitations & Future Work

This study has noteworthy limitations that may pose potential threats to its validity. First, the study involved only 31 students from a single institution. This limitation underscores the need to examine the impact of Kritik360 at a larger scale, including courses with higher student enrollments and, ideally, across multiple institutions, to strengthen the validity of these findings. Second, it was initially intended to include a control group using the other three sections of this introductory programming course in order to conduct a comparative analysis of Kritik360's impact on students' programming comprehension skills. However, the survey response rate from those sections was minimal. The objective would have been to compare student performance across all five sections and determine whether students exposed to Kritik360 exhibited stronger programming comprehension practices than those in the other sections.

As this initiative is expanded in upcoming semesters, plans are being discussed to incorporate a comparative study that includes an well-established control group. Additionally, the first and second authors are working closely with the third author and the Kritik team to establish additional in-class initiatives aimed at more rigorously examining the impact of Kritik360 on student learning outcomes in introductory programming at this minority-serving institution.

7. Conclusion

Developing effective strategies for teaching novice programmers continues to evolve, particularly with the growing presence of AI. Within the context of computing education and programming pedagogy, this expansion introduces potential concerns related to computational skill development and proficiency. However, when AI is thoughtfully integrated into the computing classroom while upholding academic integrity and ethical practices, it can serve as an additional

tool to support students' computational learning. This article highlights this potential through the adoption of Kritik360 in an introductory programming course, where the platform facilitated peer-to-peer learning as a means to support novice programmers in developing their programming comprehension skills.

References

[1] L. E. Winslow, "Programming pedagogy—a psychological overview," *SIGCSE Bulletin*, vol. 28, no. 3, pp. 17–22, 1996.

[2] B. Adelson, "When novices surpass experts: The difficulty of a task may increase with expertise," *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 10, no. 3, pp. 483–495, 1984, reprinted in Human Factors in Software Development (2nd ed.); Bill Curtis (ed.), IEEE Computer Society Press, 1985, pp. 55–67.

[3] R. Garcia, K. Falkner, and R. Vivian, "Instructional framework for cs1 question activities," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 2019, pp. 189–195.

[4] D. Reed, "Critical thinking and modeling in cs0: the prisoner's dilemma," *Journal of Computing Sciences in Colleges*, vol. 26, no. 5, pp. 197–204, 2011.

[5] T. Thomas, T. Davis, and A. Kazlauskas, "Embedding critical thinking in is curricula," *Journal of Information Technology Education: Research*, vol. 6, no. 1, pp. 327–346, 2007.

[6] S. I. Malik, R. Mathew, A. Al-Sideiri, J. Jabbar, R. Al-Nuaimi, and R. M. Tawafak, "Enhancing problem-solving skills of novice programmers in an introductory programming course," *Computer Applications in Engineering Education*, vol. 30, no. 1, pp. 174–194, 2022.

[7] K. Kwon, "Novice programmer's misconception of programming reflected on problem-solving plans." *International Journal of Computer Science Education in Schools*, vol. 1, no. 4, p. n4, 2017.

[8] C. D. Hundhausen, A. Agrawal, and P. Agarwal, "Talking about code: Integrating pedagogical code reviews into early computing courses," *ACM Transactions on Computing Education (TOCE)*, vol. 13, no. 3, pp. 1–28, 2013.

[9] C. Hundhausen, A. Agrawal, D. Fairbrother, and M. Trevisan, "Integrating pedagogical code reviews into a cs 1 course: an empirical study," *ACM SIGCSE Bulletin*, vol. 41, no. 1, pp. 291–295, 2009.

[10] R. Budhiraja, I. Joshi, J. S. Challa, H. D. Akolekar, and D. Kumar, ""it's not like jarvis, but it's pretty close!"-examining chatgpt's usage among undergraduate students in computer science," in *Proceedings of the 26th Australasian Computing Education Conference*, 2024, pp. 124–133.

[11] L. Kloub and A. Gupta, "Chatgpt in computer science education: exploring benefits, challenges, and ethical considerations," in *2024 ASEE North East Section*, 2024.

[12] F. Ouyang, L. Zheng, and P. Jiao, "Artificial intelligence in online higher education: A systematic review of empirical research from 2011 to 2020," *Education and Information Technologies*, vol. 27, no. 6, pp. 7893–7925, 2022.

[13] L. Cai, M. M. Msafiri, and D. Kangwa, "Exploring the impact of integrating AI tools in higher education using the Zone of Proximal Development," *Education and Information Technologies*, vol. 30, no. 6, pp. 7191–7264, 2025.

[14] M. Madanchian, G. Drazenovic, S. R. Ramzani, and H. Taherdoost, "Integrating AI tools to enhance learning outcomes in modern education systems," *Procedia Computer Science*, vol. 263, pp. 514–521, 2025.

[15] B. Dwiantoro, Y. Sujana, and P. Hatta, "The role of artificial intelligence in programming education and its impact on the learning process," *IJIE (Indonesian Journal of Informatics Education)*, vol. 9, no. 1, pp. 53–61, 2025.

[16] M. Amoozadeh, D. Daniels, D. Nam, A. Kumar, S. Chen, M. Hilton, S. Srinivasa Ragavan, and M. A. Alipour, "Trust in generative AI among students: An exploratory study," in *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. New York, NY: ACM, 2024, pp. 67–73.

[17] M. Körber, "Theoretical considerations and development of a questionnaire to measure trust in automation," in *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018)*. Geneva, Switzerland: International Ergonomics Association, 2018, pp. 13–30.

[18] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, "Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE 2023)*. New York, NY: ACM, 2023, pp. 500–506.

[19] S. Lau and P. J. Guo, "From "Ban it till we understand it" to "Resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot," in *Proceedings of the 2023 ACM Conference on International Computing Education Research (ICER '23)*. New York, NY: ACM, 2023, pp. 106–121.

[20] B. Xie, D. Loksa, G. L. Nelson, M. J. Davidson, D. Dong, H. Kwik, A. Hui Tan, L. Hwa, M. Li, and A. J. Ko, "A theory of instruction for introductory programming skills," *Computer Science Education*, vol. 29, no. 2–3, pp. 205–253, 2019.

[21] P. Gou and X. Ren, "Developing students' computational thinking through project-based learning empowered by generative ai: A practical study," in *Proceedings of the 2025 International Conference on Education, Knowledge and Information Management (ICEKIM 2025)*. New York, NY, USA: ACM, 2025, pp. 1–6.

[22] E. Benedetti, "Supporting teaching and learning computational thinking skills with generative ai in computing education," in *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2025)*. New York, NY, USA: ACM, 2025, pp. 1–2.

[23] S. Suh and P. An, "Leveraging generative conversational ai to develop a creative learning environment for computational thinking," in *Companion Proceedings of the 27th International Conference on Intelligent User Interfaces (IUI '22 Companion)*. New York, NY, USA: ACM, 2022, pp. 73–76.

[24] P. Haindl and G. Weinberger, "Students' experiences of using chatgpt in an undergraduate programming course," *IEEE Access*, vol. 12, pp. 43 519–43 536, 2024.

[25] R. Yilmaz and F. G. Karaoglan Yilmaz, "The effect of generative artificial intelligence based tool use on students' computational thinking skills, programming self-efficacy, and motivation," *Computers and Education: Artificial Intelligence*, vol. 4, p. 100147, 2023.

[26] S. Tian, "The integration of computational thinking and artificial intelligence serves to enhance the cognitive processes and skill acquisition of students," in *Proceedings of the 2024 International Symposium on Artificial Intelligence for Education (ISAIE 2024)*. New York, NY, USA: ACM, 2024, pp. 1–4.

[27] X. Weng, H. Ye, Y. Dai, and O.-L. Ng, "Integrating artificial intelligence and computational thinking in educational contexts: A systematic review of instructional design and student learning outcomes," *Journal of Educational Computing Research*, vol. 62, no. 6, pp. 1420–1450, 2024.

[28] K. S. Double, J. A. McGrane, and T. N. Hopfenbeck, "The impact of peer assessment on academic performance: A meta-analysis of control group studies," *Educational Psychology Review*, vol. 32, no. 2, pp. 481–509, 2020.

[29] E. Panadero, M. Alqassab, J. Fernández Ruiz, and J. C. Ocampo, "A systematic review on peer assessment: intrapersonal and interpersonal factors," *Assessment & Evaluation in Higher Education*, vol. 48, no. 8, pp. 1053–1075, 2023.

[30] J. Matejić, "Students' perceptions of peer assessment in a web programming course: Reflections on feedback, learning, and professional improvement," *Journal of Educational Studies in Mathematics and Computer Science*, vol. 2, no. 1, pp. 56–66, 2025.

[31] J. R. Power and D. Tanner, "Peer assessment, self-assessment, and resultant feedback: an examination of feasibility and reliability," *European Journal of Engineering Education*, vol. 48, no. 4, pp. 615–628, 2023.

[32] M. H. Ismail, J. J. Sijore, S. Ismail, A. H. A. Hamid, and M. N. A. Nor'a, "A systematic literature review on recent peer code review implementation in education," in *Proceedings of the 2024 International Conference on TVET Excellence & Development (ICTeD 2024)*. IEEE, 2024.

[33] A. Y. Wang, Y. Chen, J. J. Y. Chung, C. Brooks, and S. Oney, "PuzzleMe: Leveraging peer assessment for in-class programming exercises," *Proceedings of the ACM on Human-Computer Interaction*, vol. 5, no. CSCW2, p. Article 415, 2021.

[34] A. K. Al-Khalifa and M. Devlin, "Evaluating a peer assessment approach in introductory programming courses," in *Proceedings of the United Kingdom & Ireland Computing Education Research Conference (UKICER 2020)*. Association for Computing Machinery, 2020, pp. 51–58.

[35] T. Brown, M. R. Narasareddygari, M. Singh, and G. Walia, "Using peer code review to support pedagogy in an introductory computer programming course," in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–7.

[36] S. Berrezueta-Guzman, S. Krusche, and S. Wagner, "From coders to critics: Empowering students through peer assessment in the age of AI copilots," 2025.

[37] K. J. Stylianides, B. A. Greenberg, G. M. Tarud, E. A. Huck, C. Cruz, and L. Cruz, "Everyone's a Kritik™-student perceptions of peer feedback as a social learning tool," *College Teaching*, pp. 1–10, 2025.

[38] Kritik Help Center, "What is Kritik?" n.d. [Online]. Available: https://help.kritik.io/en/articles/6387869-what-is-kritik

[39] ——, "Participating in a Kritik activity," n.d. [Online]. Available: https://help.kritik.io/en/articles/6387956-participating-in-a-kritik-activity

[40] ——, "Creating rubrics with AI," n.d. [Online]. Available: https://help.kritik.io/en/articles/8853473-creating-rubrics-with-ai

[41] ——, "Spotlight feature," n.d. [Online]. Available: https://help.kritik.io/en/articles/6387537-spotlight-feature

[42] ——, "How scoring works," n.d. [Online]. Available: https://help.kritik.io/en/articles/6388500-how-scoring-works

[43] ——, "Grading power and score calculation in Kritik," n.d. [Online]. Available: https://help.kritik.io/en/articles/6845495-grading-power-and-score-calculation-in-kritik

[44] ——, "VisibleAI + Kritik360 bundle syllabus insert," n.d. [Online]. Available: https://help.kritik.io/en/articles/12023013-visibleai-kritik360-bundle-syllabus-insert

[45] H. Khosravi, S. B. Shum, G. Chen, C. Conati, Y. S. Tsai, J. Kay, others, and D. Gašević, "Explainable artificial intelligence in education," *Computers and Education: Artificial Intelligence*, vol. 3, p. 100074, 2022.

[46] A. K. Sofjan, K. A. Nguyen, D. Surati, and C. Marette, "Assessment of the validity of peer scores and peer feedback in an online peer assessment platform (Kritik)," *Currents in Pharmacy Teaching and Learning*, vol. 15, no. 4, pp. 400–407, 2023.

[47] V. Braun and V. Clarke, *Thematic analysis.* American Psychological Association, 2012.