PostCommit: GenAI-Powered Code-to-Concept Assessment for Computer Science Students

Dr. Andrew Forney, Loyola Marymount University

Prof. Andrew Forney is an Associate Professor of Computer Science at Loyola Marymount University with research interests broadly at the intersection of cognitive psychology, artificial intelligence, and experimental design. Although his formal specialty is in the areas of Causal and Counterfactual Inference, his passion for education attempts to blend modern AI tools and analyses for the betterment of the student experience.

Aidan Srouji, Loyola Marymount University
Chela Willey, Loyola Marymount University

PostCommit: GenAI-Powered Code-to-Concept Assessment for Computer Science Students

Abstract

Programmatic assignments compose large portions of a student's responsibility in many Computer Science (CS) curricula, aiming to cement and assess Code-to-Concept (C2C) mapping. However, in the era of Generative AI (GenAI), CS students may complete assignments without mastering any C2C proficiency, leaving valuable knowledge units behind that will be demanded in subsequent, more complex work. In response, we propose a new class of GenAI application for Assessment Assistance (GenAI-AA) and feature one implementation through an app called PostCommit: a novel tool for instructors to conduct individualized technical interview style code questions asking students to explain in plain-language portions of their submitted work. By providing a second round of assessment for C2C proficiency in a proctored setting (accomplished through the PostCommit interface), students gain a new motivation for deep C2C understanding, regardless of the sources used to complete their programmatic assignments, and supplemental practice explaining their own ideation and understanding of submitted code (a skill that is often omitted from CS curricula). Likewise, instructors gain a new tool for assessing their class' C2C mastery and providing individualized feedback on how students can improve their explanations in interview or conference settings. This work details the PostCommit workflow from student code submission via GitHub Classroom to quiz question curation and automated feedback. We also present pilot data from 6 CS courses spanning grade levels of the CS curriculum that incorporated PostCommit across an academic semester, including comparisons of student PostCommit quiz scores and performance on other course elements (i.e., assignments and exams), rates of instructor changes to automated feedback, and overall student perception of PostCommit effectiveness. Data suggest large gaps in student C2C mastery when comparing explanatory vs. programmatic correctness, with PostCommit scores correlating with exam, but not assignment performance. These results highlight a new need for motivating understanding, rather than simply completion, of assignments: a priority that GenAI-AA tools may be used to meet in the adjacent future.

Introduction

Generative Artificial Intelligence (GenAI) has seen a meteoric rise in applications within many domains with models' abilities for image, video, and text production, summarization, and translation assisting lawyers analyzing cases, designers in their ideation, programmers writing code, and many more [1]. Although the GenAI landscape is vast and ever-expanding, this work focuses on two main facets relating to Computer Science:

- *GenAI for Productivity Assistance (GenAI-PA):* among their other capabilities, most of the foundational large language model (LLM) providers (e.g., OpenAI's ChatGPT [2], Google's Gemini [3], and Anthropic's Claude [4]) have positioned themselves as

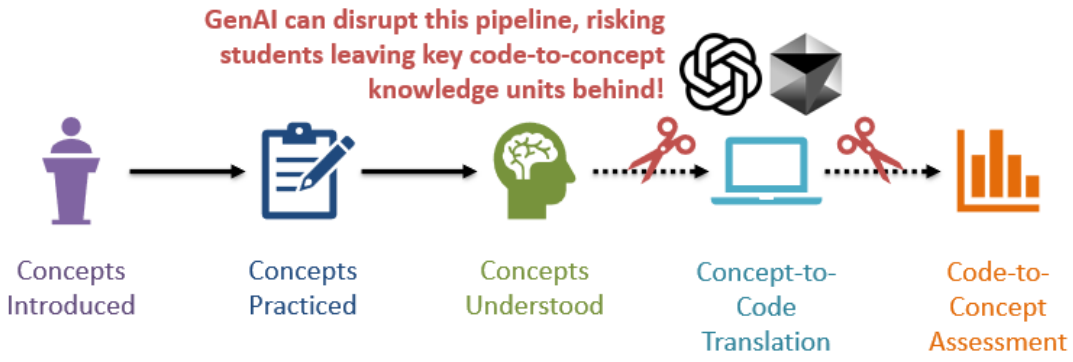

Figure 1. The prototypical CS educational workflow and where GenAI may disrupt it.

computer programming assistants capable of producing, debugging, documenting, and helping to interpret code, with specialized software like Anysphere's Cursor [5] providing an integrated development environment that can harness the above.

- *GenAI for Learning Assistance (GenAI-LA):* although these major GenAI-PA providers have likewise pitched applications as learning assistants, such as by serving as interactive instructors or quiz generators [6] or use as programming tutors [7], related apps have explored turning technical content into accessible podcasts (e.g., Google's NotebookLM [8]), "deep-research" tools that can find / summarize prompt-driven sources and articles [9], and many others [10].

These rapidly developed tools that have been suddenly placed in the hands of practitioners and students alike have been met with mixed results and perspectives. A recent survey of undergraduate GenAI use had $1/4$ of students self-report using it to complete full assignments for them (citing primarily pressure to obtain good grades, time constraints, apathy towards academic honesty policies, and social-normative belief that their peers are doing the same), with roughly $1/2$ cautious that these tools can compromise critical thinking [11].

Classroom *instructors*, on the other hand, may feel that their educational objectives are at odds with both GenAI-PA and GenAI-LA, which may enable critical knowledge units to be missed by students who abuse these as shortcuts to learning (either knowingly or not) [12], [13]. This is not a new problem, as instructors have long bemoaned the availability of test/answer banks, illicit external assistance with assignments (e.g., friends who have previously taken the course or essay-writing services), etc., but it is *exacerbated* by the ubiquity and ease of GenAI as well as the difficulty in detecting GenAI-products from genuine student artifacts [14].

Educators in Computer Science thus find themselves in the difficult position of Fig. 1: the industry for which they are preparing students is embracing GenAI-PA tools, and so educators may feel obligated to prepare students to learn these tools while in school [7]. Yet, those same tools may allow students to appear to be thriving in the classroom when students have, in fact, extracted very few of the intended knowledge units from their assignments [15]. This challenge is complicated further by the difficulty in quantifying the degree to which GenAI may be helping or harming a student's *understanding* and ability to translate concepts from lecture to the code they produce rather than simply improving the efficiency of their production [16],

[17], [18]. On the other hand, proctored settings like exams generally do not assess the same C2C abilities that at-home programming demands, since the latter take ample time to think, code, debug, test, etc. that cannot be accomplished in the span of an exam window.

Herein, we propose an additional class of tools to the GenAI educational ecosystem to complement GenAI-PA and GenAI-LA: *GenAI for Assessment Assistance (GenAI-AA).* The primary goal of GenAI-AA is twofold: (1) to help students calibrate the extent to which their C2C mapping has been helped, harmed, or left wanting by GenAI, external assistance, or other potential impediments to learning, and (2) to provide educators with a tool to more accurately assess whether or not the intended knowledge units were conveyed by the *process* of an exercise, rather than its *final product.*

This paper demonstrates *one possible approach* to GenAI-AA that recognizes that some assignments require the time, attention, and iterative development best suited for take-home assessment, but novelly provides a second round of "assessing-the-assessment" to ensure student authorship and understanding. Concretely, we have developed and deployed a webapp called PostCommit that harnesses GenAI to create instructor-curated technical-interview-style questions about a student's *submitted code* and delivers these questions through a web-browser interface that allows them to be administered in an in-class, proctored setting. What follows are the details surrounding PostCommit's construction and analysis of its pilot data from several CS undergraduate courses; specifically, this paper provides:

1) An overview of the PostCommit Instructor and Student workflow as an example of GenAI-AA.
2) A comparison of student PostCommit quiz scores vs. their programmatic assignment correctness scores demonstrating large gaps in C2C proficiencies.
3) An examination of instructor usage statistics and interactions with the PostCommit question-generation and grading system.
4) Data-driven recommendations and motivations for integration of GenAI-AA into curricula both within and beyond computer science.

PostCommit Overview

The GenAI-AA feedback loop intended by the PostCommit app (Fig. 2) attempts to stem the loss of knowledge risked through GenAI-PA tools (Fig. 1). This section demonstrates the app's interface that accomplishes this workflow from both the instructor's and students' perspective.

For roster and code repository management, PostCommit is built atop GitHub Classroom (GHC) [19], giving students practice using industry-standard version control while integrating with traditional CS curricula learning tools. As such, *all* users (both instructors and students) require a GitHub account and to own / be enrolled in at least one GHC.

PostCommit Instructor Interface

Instructors begin their workflow by authenticating with GitHub, upon which they are greeted by the **Instructor Dashboard** containing all GHC classes that they both own and have linked

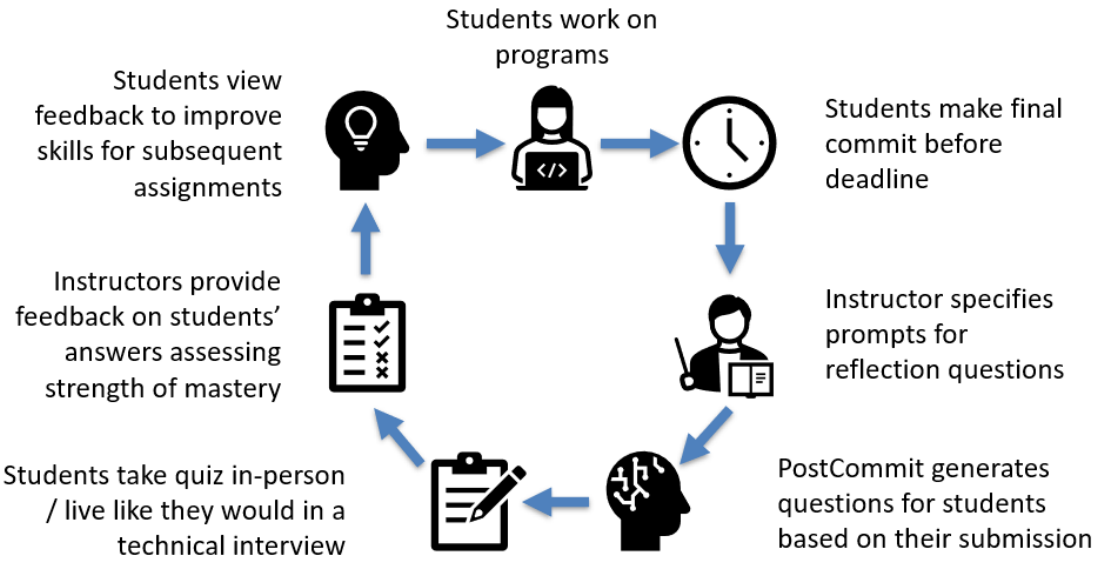

Figure 2. The PostCommit feedback loop insofar as it pertains to the sequence of a CS course's assignments.

to PostCommit. Clicking into any one of the linked classes yields the **Classroom Dashboard** through which the class' roster, assignments, and assessments can be managed.

Each class' **Roster View** shows all enrolled students who have successfully linked their GitHub accounts, with the ability to tunnel into any student's details and, most notably, view the repositories associated with their submissions and grant additional time on the PostCommit quizzes (as might be the case for students with testing accommodations).

Upon adding a GHC assignment, instructors are able to use the Assessment interface to create a corresponding **PostCommit Quiz (PCQ)**, deciding essential characteristics like the base amount of time students have to complete the quiz. This time can be anything from a 10-minute in-class assessment to days specifying how long students have to complete the assignment at-home.

Following these essential Assignment characteristics, instructors will specify their intended PCQ Question-Generation (QGen) process, as diagrammed in Fig. 3. The QGen process begins with instructors selecting one or more context file directories (e.g. `./src/pathfinder.py`) within each student's GHC repositories containing the code / content from which they wish to create PCQ questions. Context files can be any text-based file (source code, specs, README files, reports, etc.) that will be appended as context to the GenAI PCQ QGen process. This is done individually for each student's repository, so questions are personalized to the student's submission.

Instructors may then specify one or more question prompts through which the QGen process will create PCQ questions, an example of which is shown on a sample A* pathfinder assignment in Fig. 4. Anecdotally, pilot use has revealed a successful pattern of crafting these prompts in a 3-step process:

1) *Find:* Specify what part of the student's submission to reference / where the question context is likely to occur.

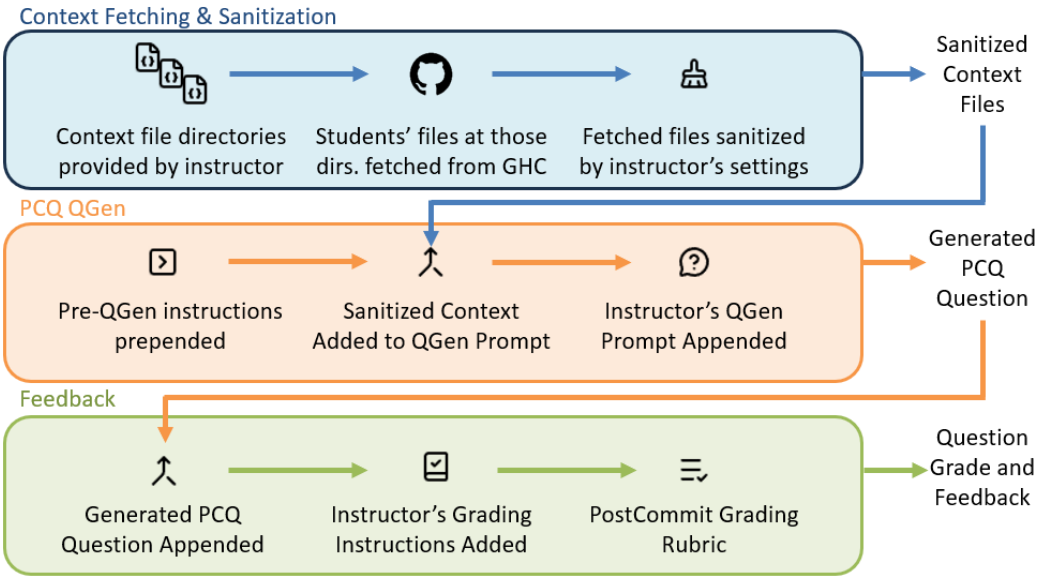

Figure 3. PCQ QGen and Feedback workflows.

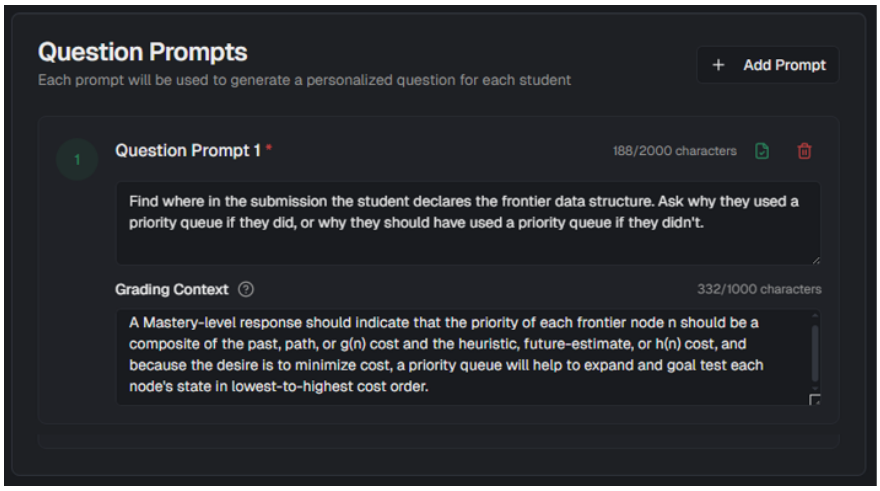

Figure 4. Example PCQ question-generation prompt on an A* pathfinding assignment.

2) *Ask:* Determine the relevant C2C topic to ask students to explain as would be manifest in their submitted code.

3) *Expect Exceptions:* Add clauses to specify how to ask a question for a student who has chosen an alternative / missing implementation if the expected concept is not found.

Finally, instructors may choose from amongst several settings parameterizing the QGen process and options for the administering of the PCQ (Fig. 5). Notably, basic security settings can be employed for in-person, proctored PCQs like ensuring that students take the quiz in full-screen mode (and logging if they ever leave) and pruning student comments from the displayed code (so that students cannot simply leave themselves detailed notes with how to answer PCQ questions). Adjusting the Question Variability parameter can ensure that questions

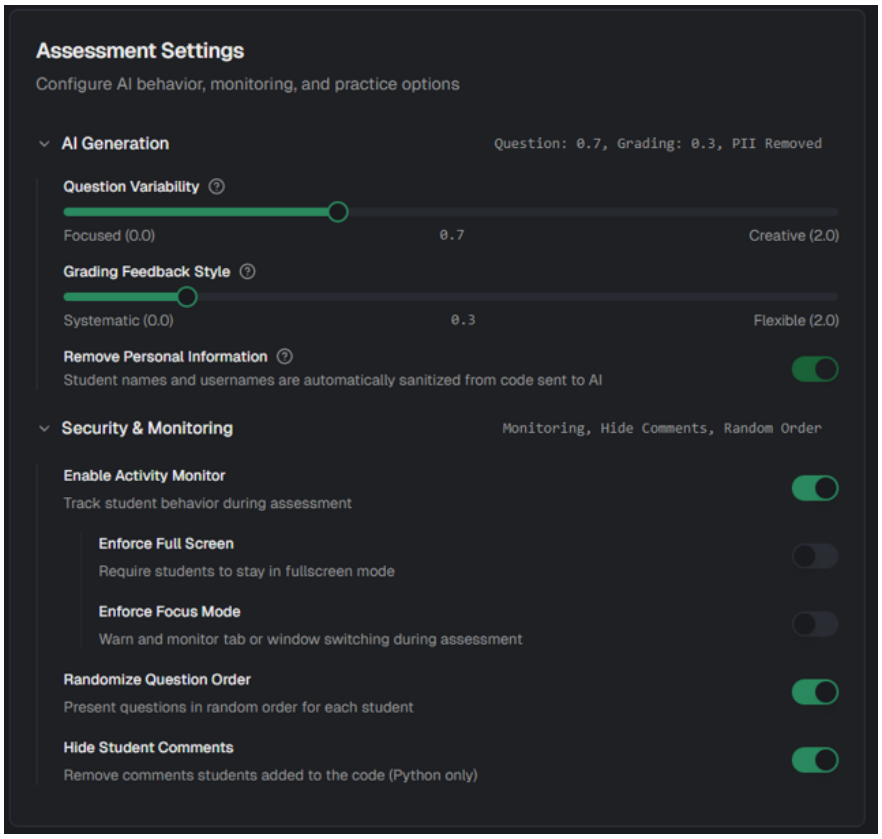

Figure 5. Settings panel for PCQ QGen process.

are more or less standardized between students, which partly accounts for normalized difficulty along with how instructors phrase their prompts. The full QGen prompt (Fig. 3) is then sent[1] to Google's Gemini [3] LLM to produce the specific PCQ questions that will be served to students during their quiz.

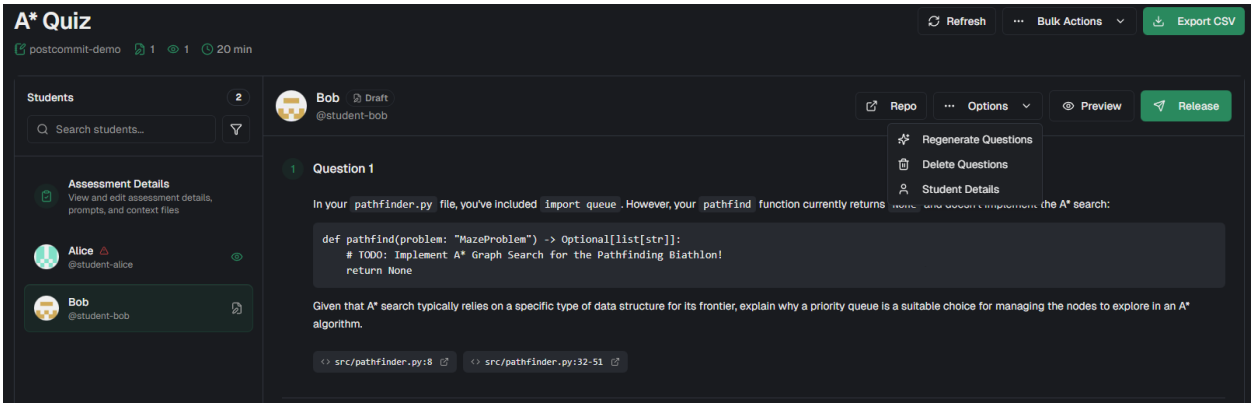

Figure 6. Example PCQ Draft question for student who did not finish the assignment.

[1]Note: all student-identifying information is first pruned from the context files and PostCommit uses the Gemini API calls that forbid training on the submitted prompt.

The results of the QGen process can be viewed by the instructor in the **Assessment's Draft Phase**, during which there are options to regenerate or edit questions, or Release the quiz to then be taken by students. Fig. 6 shows an example student's Draft of Question 1, which successfully crafts a PCQ question even though the student never finished the assignment (thus allowing for missed knowledge units to be salvaged albeit in lesser rigor).

After students have completed their PCQ (see Student Interface section below), PCQs enter the **Feedback Phase** through which instructors may assess the quality of the student's responses. Student answers are graded on a 0-2 point scale:

0) *Missing* = Student did not demonstrate any strong connection of concepts to code
1) *Gist* = Student had some of the right words but not beyond a surface level of depth or accuracy to suggest mastery
2) *Mastery* = Student explained the concept-code mapping precisely and using appropriate vocabulary

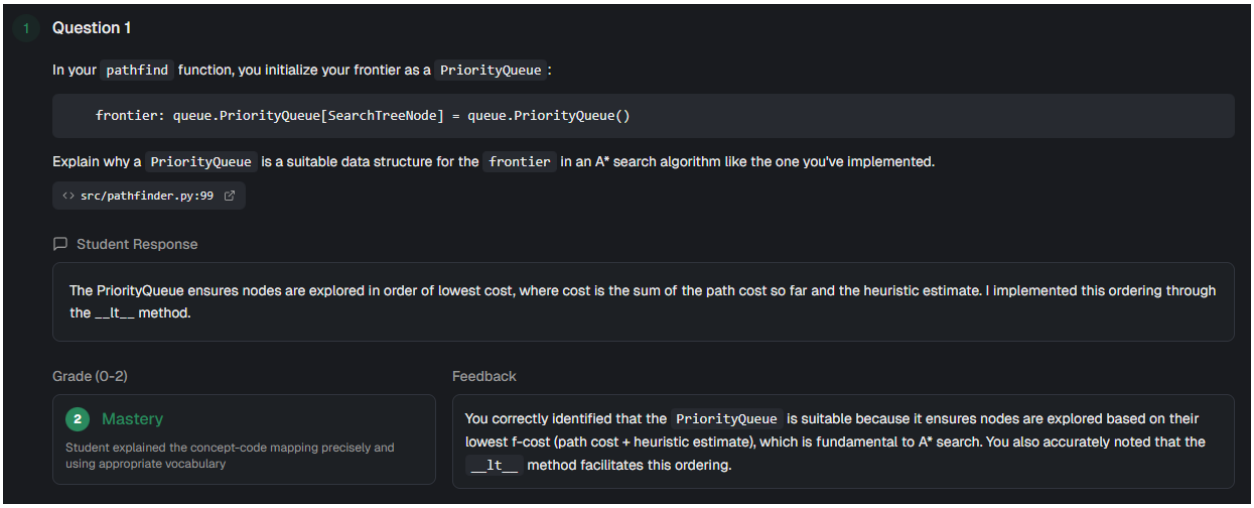

Figure 7. Example PCQ Feedback interface for student with Mastery-level response.

instructors may then employ PostCommit's Autograder functionality diagrammed at the bottom of Fig. 3 to assess and provide feedback on students' responses, which can then be modified by the instructor before moving the quiz to the Published phase wherein students may view their feedback, an example of which is in Fig. 7.

PostCommit Student Interface

PCQs can be released selectively to only certain students (e.g., if taking attendance and releasing only to those students in-class) or all at once, but once a student's PCQ has entered the Released Phase, the timer has begun and they must access and submit their PCQ through the PostCommit student interface before that time elapses.

Fig. 8 shows the Student PCQ interface while they are taking a PCQ in full-screen mode. The left panel contains each PCQ question along with a free-response textbox and a "flag" option for bringing questions to the attention of the instructor if there is an error. The right panel

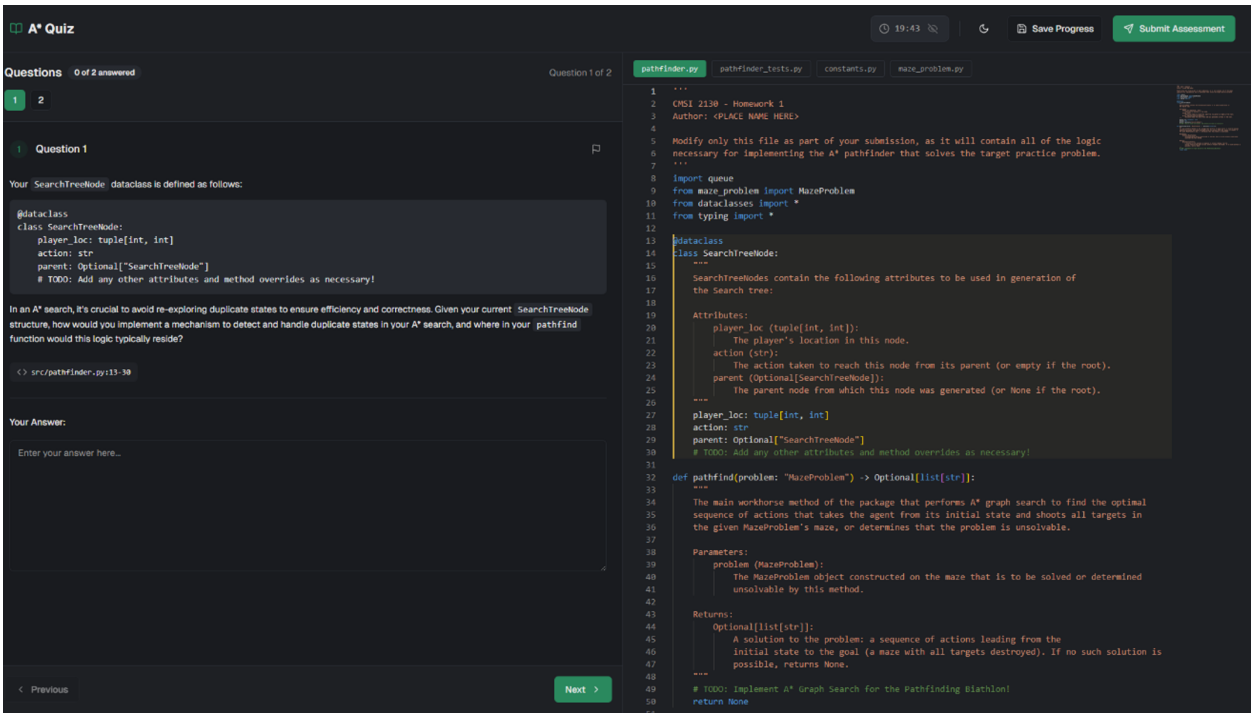

Figure 8. PCQ Student interface.

contains all of the student Context files selected by the instructor with highlighting applied to code cited by the current question.

After the PCQ reaches the Published phase (following instructor feedback), students may return to PostCommit to view their feedback in a display identical to the instructor's in Fig. 7.

Research Questions

With both student and instructor workflows of PostCommit now detailed, we transition to the empirical methods of its pilot deployment that was intended to answer the following research questions:

1) Is programmatic assignment performance still a reliable metric for C2C understanding in the era of GenAI?
2) Can GenAI-AA tools like PostCommit expose C2C mastery deficits and their relation to other assessments like exams?
3) How do students' programming-confidence-levels relate to their demonstrated C2C mastery in the era of GenAI?

Method

PostCommit pilot participation was limited to the Loyola Marymount University's Computer Science Department with pilot data collected from the Fall 2025 semester (15 weeks long). Eleven courses' instructors signed up for the pilot, but only 7 courses ultimately featured PCQ in their offering. Of these, 1 was a graduate course whose heterogeneous class composition

Table I
PARTICIPATING COURSE INFORMATION AND PCQ COUNTS IN PILOT STUDY. INSTRUCTORS OF EACH COURSE ARE REFERRED-TO BY ANONYMIZING LETTERS A, B, AND C. INSTRUCTOR C IS AN AUTHOR ON THIS PAPER.

Course	Level	Language	Participants	Instructor	N PCQs
Data Structures (DS)	SO	Java	26	A	3
Algorithms (Algs)	SO	Python	41	B	4
Introductory Artificial Intelligence (AI)	JR/SR	Python	43	C	5
Cognitive Systems Design (CogSys)	JR/SR	Python	27	C	5

(undergraduates could also take the course) and grade computations rendered it incomparable to the others. We herein focus on the 6 undergraduate courses whose final tally included 4 different courses, 2 with 2 sections each, between 3 separate instructors (1 of which authored this paper). A summary of these courses is given in Table I with Participant Enrollment counts only tallying those who consented to participate in the PostCommit study (a total of 23 students declined participation and are thus omitted from the data presented herein).

Participants

Participants were a total of 137 Undergraduate students enrolled in both lower-division CS foundations courses (67 within Data structures and Algorithms) and upper-division AI courses (70 within Artificial Intelligence and Cognitive Systems Design [a blend of reinforcement learning and causal inference]). Seven students did not complete the majority of the PCQs and homework assignments and were excluded from analyses. Each participating course employed PCQs as 10% of the course's final grade, but all students (participating in the research component or not) received full credit on each PCQ as long as they were present in-class and completed the quiz. However, to motivate the quality of their responses to the PCQs, participants were offered 1 raffle ticket into an end-of-semester drawing for 2x$50 Amazon gift cards per course just for signing up, and a bonus raffle ticket for every PCQ question answered with Mastery-level quality. Participants also received 1 raffle ticket for completing an end-of-semester survey on their experience. All human-subjects procedures that follow have Institutional Review Board (IRB) approval.

Procedure

Throughout the Fall 2025 semester, instructors in each participating class would host a PCQ in the next in-person class meeting following a programmatic assignment's deadline, as constructed via the workflow detailed in the PostCommit Overview, with the exception of the first, which was given online for students to become calibrated to the PCQ interface and ensure its compatibility with their personal devices. Each PCQ was between 3-5 questions with in-class quizzes being allocated 10 minutes[2] for completion, with the exception of the CogSys final project PCQ, which was a 5-question quiz allotted 20 minutes.

The procedure for PCQ administration was as follows:

1) Sometime following the assignment's deadline, instructors compose each student's individualized PCQ using the QGen pipeline, vetting generated questions.

[2]Those with accommodations were granted the extra time to complete these, e.g., a student with $+50\%$ time could spend up to 15 minutes on the 10-minute quizzes.

2) On the first take-home PCQ, students would link their GHC account with the PostCommit app and then digitally view and decline / accept the Informed Consent for the study's research component as part of the app's registration. Participation required that students be 18 years of age or older.
3) When a PCQ was administered in-person, instructors would have students open their PostCommit student dashboard and await the PCQ to be released. Only students present in-class would have their PCQ released via roll-call.
4) Students would then complete the PCQ in the time allotted. Following its conclusion, students were given an optional survey asking about their confidence on their answers and with the PCQ's material as well as sources of help that they had on the assignment.
5) Following the in-class administration, instructors graded and provided PCQ feedback to students, who could then view their results via the app.

Following the semester's conclusion, student participants volunteered their course assignment and exam grades for the analysis that follows.

Analysis

Correlative analysis was performed between the following main assessment metrics:

- $PCQ\%$, the percentage of PCQ quality points divided by the total, e.g., a student who received one Mastery-level (2-points) and one Gist-level (1-point) response on a 2-question PCQ would obtain a $PCQ\% = 3/4 = 75\%$.
- $Assignment\%$, the percentage of programmatic homework assignment *correctness* earned by the student's submission, as judged by unit test completion, Python `mypy` compliance, documentation, programmatic style, etc.[3]
- $Exam\%$, the percentage of points earned on each exam querying primarily the conceptual (not programmatic) topics of the course.[4]
- $ConfidenceLevel$, student self-report confidence levels on their "understanding of the code they submitted" for the assignment (sampled directly following PCQ submission and before graded feedback), ranging from 1 = not understood at all to 5 = fully understood.

From these, we derived one additional metric examining the difference between PCQ performance and Assignment correctness: $\Delta PC = PCQ\% - Assignment\%$. Students with large, negative ΔPC values were those who likely had an abundance of outside-assistance, allowing them to reach high programmatic correctness but little evidence of extracted knowledge units.

Results

Only 1 instructor for 2 classes (N = 70 students) made edits to the grades and feedback produced by the QGen process, likely because pilot testing of the QGen process was

[3]DS and Algs. assignments are completed as individuals, but AI and CogSys assignments could be completed in groups of up to 3 students.

[4]DS, Algs, and AI all feature non-cumulative midterms and finals, whereas CogSys had no exams, but a comprehensive final-project that is more comparable to the others' programmatic assignments.

ultimately graded for completion in all classes. Out of the 10 PCQs administered (5 per class), 1 PCQ contained 5 questions, 2 contained 4 questions, and 7 contained 3 questions, resulting in 1,190 possible questions across both classes for which edits could be made. Across the two courses, 50 grade edits (across 45 unique PCQs) and 60 edits (across 53 PCQs) were made to written feedback on individual questions. This amounts to 4.20% question grades and 5.04% question feedback were edited. Each question was worth 2 points. The average grade change for the 45 PCQs that contained grade edits was an increase of 1.2 points.

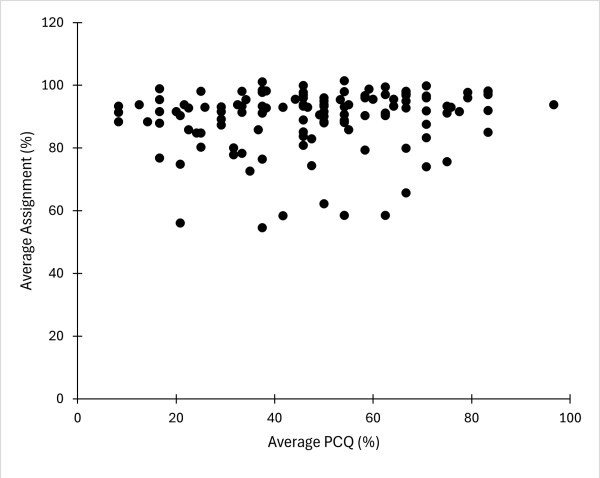

Figure 9. Average $assignment\%$ and average $PCQ\%$ across courses

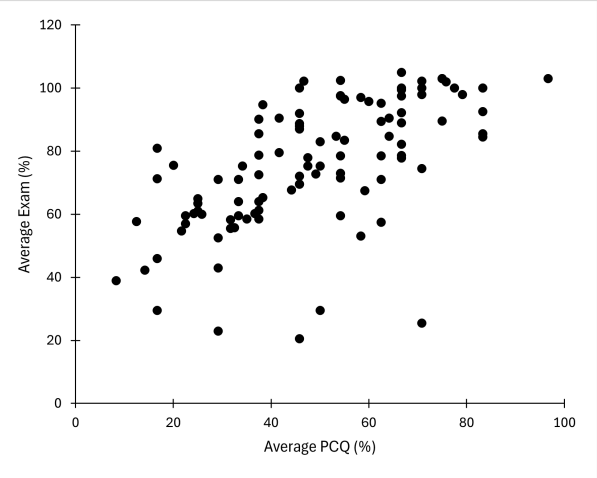

Figure 10. Average $exam\%$ and average $PCQ\%$ across courses.

We performed Pearson's r correlation tests between $PCQ\%$, $Assignment\%$, and $Exam\%$. We found that $Assignment\%$ ($M = 89.60\%, SD = 9.69$) was positively correlated with $Exam\%$ ($M = 75.64\%, SD = 20.10$), $r(101) = .267, p = .006$, but not with $PCQ\%$ ($M = 48.37\%, SD = 19.54$), $r(128) = .147, p = .095$, Fig. 9. Importantly, $Exam\%$ was significantly positively correlated with $PCQ\%$, $r(101) = .613, p < .001$, Fig. 10.

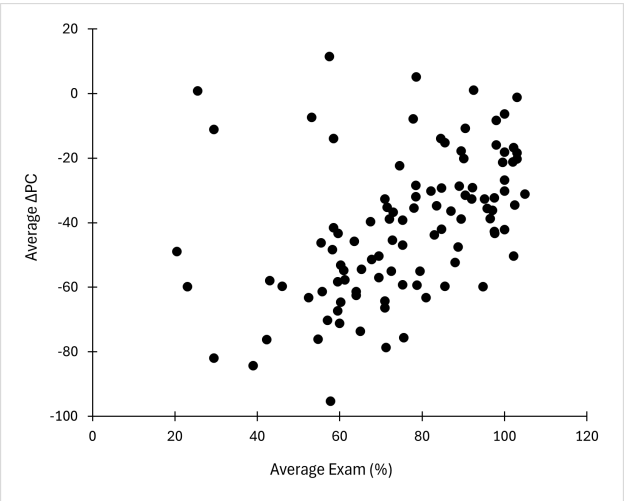

Figure 11. Average exam percent and average difference between assignment percent and PCQ percent per assignment.

We performed a Pearson's r correlation test between mean $Exam\%$ and ΔPC ($M = -41.06\%, SD = 21.92$) to test whether those who had greater differences between their ability to demonstrate their conceptual knowledge tested in the PCQ and their assignment correctness had lower overall in-person exam scores. We found that those with more negative ΔPC values (lower PCQ scores than their corresponding assignment scores) were significantly correlated with lower exam scores, $r(101) = .427, p < .001$, Fig. 11.

Table II
MEAN $PCQ\%$ PERCENT BY CONFIDENCE LEVEL

Confidence Level	Mean Percent	SD Percent	Count
1	8.33	11.79	2
2	36.25	32.81	10
3	45.87	23.51	111
4	60.60	23.67	171
5	76.98	21.14	82
Missing	30.03	30.39	194

Tab. II compares mean $PCQ\%$ across self-reported confidence levels across all PCQ assessments and students. A one-way ANOVA suggested that $PCQ\%$ differed significantly between self-reported confidence levels, $F(4, 371) = 25.89, p < .001$, such that higher scores was associated with more confidence in the material.

Twenty-nine students across all courses completed an end of the semester survey about their views of the PCQ. Overall, students viewed the PCQ, and the feedback they received favorably. The results can be seen in Tab. III.

Table III
MEAN LIKERT RATINGS FOR PCQ SURVEY STATEMENTS

Statement	**Mean**	**SD**
PostCommit quizzes motivated me to better understand my code.	4.07	0.80
PostCommit quiz feedback helped me improve the way I answer technical interview questions.	3.86	0.99
I felt that the scores I received on my PostCommit quiz answers were fair and well-reasoned.	4.03	0.98
With sufficient studying, I feel that I could master every question on my PostCommit quizzes.	4.17	0.97
PostCommit increased my confidence in explaining code or computing concepts.	3.79	1.05

Note. Ratings were made on a 5-point Likert scale (1 = Strongly disagree, 5 = Strongly agree).

Discussion

[RQ1] Is programmatic assignment performance still a reliable metric for C2C understanding in the era of GenAI? Fig. 9 alone warrants some alarm with respect to this RQ, demonstrating that students across courses have a high average $Assignment\%$ ($M = 89.60\%$), but an almost uncorrelated $PCQ\%$ ($M = 48.37\%$). This suggests that students are submitting largely functional code that they can only sparsely explain, the reasons for which could be any number of those outlined in the introduction. The proximity of each PCQ to the assignment's deadline is so brief (2-5 days) that it would seem improbable that students are forgetting its lessons in that time span, or their understanding is so ephemeral that educators may have greater concerns than the above.

[RQ2] Can GenAI-AA tools like PostCommit expose C2C mastery deficits and their relation to other assessments like exams? Given that PCQ questions query a student's own submitted code

on the concepts that would have been needed to successfully implement it, and that this assessment is so powerfully correlated to assessments of the conceptual understanding in exams (Fig. 10), we assert that GenAI-AA tools like PostCommit can indeed expose mastery deficits. This is further evidenced by the ΔPC correlation with exams in Fig. 11, suggesting that those with highly negative ΔPC may be the most at risk of missed knowledge units from, and/or misuse of external assistance on, assignments.

[RQ3] How do students' programming-confidence-levels relate to their demonstrated C2C mastery in the era of GenAI? There are two remarks from the self-reported confidence data presented in Tab. II: (1) student confidence does appear to be good predictors in their PCQ performance, demonstrating an important metacognitive capability, and (2) very few students reported ever having low confidence (≤ 2) in their understanding; rather, it seems that, based on those who elected *not* to report their confidence (i.e., the Missing value) had corresponding PCQ performance comparable to those in the 1-2 range. This reluctance to confess a lack of understanding, even though the PCQ questions' grades / feedback did not affect the student's course grade, has implications for fostering growth mindsets in students and helping them to practice metacognitive reflection, especially if this reluctance is due to shame from GenAI-use or other illicit external assistance. In future iterations, it may be useful to assess students' confidence levels in conceptual understanding of their programmatic assignment submissions *before* completing the PCQ.

In the bigger-picture of the assessment landscape, we argue that the above has exposed a blindspot: if we assume that assignments assess practical skills and exams conceptual skills, educators in the era of GenAI may owe a renewed focus on the bridge between practicum and conceptual mastery. At the very least, data like in Fig. 9 should give educators pause to consider what assignment performance tells them about the levels of understanding of material that their students have achieved. GenAI-AA may feature in this pursuit as it is fast (verifying QGen questions takes substantially less time than hand-crafting them, especially given their individualized nature), scalable (the alternative being something like oral interviews on students' code, which is unsustainable for large classes, many assignments, and may be biased), and inexpensive (the total Gemini charges for this entire pilot was $\approx \$6$ USD).

It is also worth positioning the value of these tools to students outside of assessment alone: when introduced in the pilot, instructors told students that the types of questions PCQs pose may be asked of them on technical interviews and at conferences when they are sharing their work, so practice using appropriate vocabulary and technical background is to the benefit of their career. Moreover, we hazard to claim that most students stop thinking about an assignment the moment it is submitted / graded whereas tools like PostCommit may provide fruitful avenues for spaced-repetition and even correction of mistakes: e.g., one question on a data structures review assignment asked students to find keys between two dictionaries who had conflicting Boolean values. A student who inefficiently iterated over the union of keys was able to reflect via a PCQ question, saying "I could iterate through only the keys that overlap between the dictionaries..." The largely positive student-reception evidenced by the survey data in Tab. III suggests that student perception aligned with these constructive goals, though since the survey was self-selected, the data may be missing the voices of dissenters.

Limitations and Future Directions

PCQ Security: although in-person PCQs were administered in proctored settings, it is possible that students were still able to obtain outside assistance like through screen sharing, browser extensions, and other workarounds that might not have been detected by our in-app activity monitor. Still, the reward for answer quality (raffle tickets) was unlikely enough to sponsor illicit behavior like this, nor does the data / review of suspicious activity logs suggest it, but future concerns over PCQ score purity might recruit tools like LockdownBrowser [20].

Normalization of Difficulty: because PCQ questions are generated from a student's unique submission, an instructor's potentially open-ended prompt, and use GenAI tools that necessarily operate with some nondeterminism, there is some uncontrolled variance in the question difficulty posed to each student. That said, question generation temperature was kept low for all quizzes and were generated on assignments without much latitude for variation in approaches; we argue that the strong correlation of PCQ performance with exam performance mitigates the concern over large difficulty variances, and anecdotally, we found most PCQ questions generated from the same prompt to simply be rewordings of the same underlying question.

Effort Grade: students in this pilot received only participation grades for taking each PCQ; it is possible that there was a lack of extrinsic motivation to study and otherwise deepen C2C mapping in preparation for each PCQ (or to invest the cognitive energy in crafting high quality responses). Future research will examine the effects on PCQ answer quality when PCQ *performance* (rather than mere *participation*) may factor into students' course grades.

Attribution of Performance: throughout, we have posited the ubiquity of GenAI-PA tools as the most likely rationale for large, negative ΔPC values, but this study has only *identified* such students, not causally-attributed their deficits to GenAI. Another explanation that demands future investigation is the possibility of maladaptive division of labor in group assignments, wherein one student may have disproportionately worked-on, and therefore learned-from, an assignment over the other group members. Likewise, test-taking ability may be the "third-variable" explanation connecting PCQ to exam performance.

Conclusion

This work motivates and prototypes a new theme of GenAI application in education: GenAI for Assessment Assistance (GenAI-AA). Through pilot deployment of a webapp called PostCommit, we demonstrate how GenAI-AA can provide a scalable vehicle for remotivating deep understanding of submitted assignment artifacts. Although the PostCommit pilot focuses on CS course programmatic assignments, the utility of GenAI-AA to operate on any text-based artifact positions its assessment paradigm for application in adjacent disciplines as well (e.g., courses with essays or lab reports). If nothing else, data from this pilot begs educators to seriously reflect on one question for their classes: "If a student appears to be excelling on their assignments, are you sure that they have understood its lessons?"

Acknowledgments

We would like to thank Profs. Coleman, Salman, and Shang for volunteering their time in participation of this study.

References

[1] S. S. Sengar, A. B. Hasan, S. Kumar, and F. Carroll, "Generative artificial intelligence: a systematic review and applications," *Multimedia Tools and Applications*, vol. 84, no. 21, pp. 23 661–23 700, 2025.

[2] OpenAI, "Chatgpt (gpt-5.2) [large language model]," 2025, accessed 2026-01-14. [Online]. Available: https://chat.openai.com/

[3] Google, "Gemini 3 [large language model]," 2025, accessed 2026-01-14. [Online]. Available: https://gemini.google.com/

[4] Anthropic, "Claude opus 4.5 [large language model]," 2025, accessed 2026-01-14. [Online]. Available: https://claude.ai/

[5] Anysphere, "Cursor (ai coding editor)," 2025, accessed 2026-01-14. [Online]. Available: https://www.cursor.com/

[6] K. Kadaruddin, "Empowering education through generative ai: Innovative instructional strategies for tomorrow's learners," *International Journal of Business, Law, and Education*, vol. 4, no. 2, pp. 618–625, 2023.

[7] O. Petrovska, L. Clift, F. Moller, and R. Pearsall, "Incorporating generative ai into software development education," in *Proceedings of the 8th Conference on Computing Education Practice*, ser. CEP '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 37–40. [Online]. Available: https://doi.org/10.1145/3633053.3633057

[8] Google, "Notebooklm [ai research and note-taking tool]," 2026, accessed 2026-01-14. [Online]. Available: https://notebooklm.google.com/

[9] Google, "Deep research," https://deepresearch.google/, 2025, aI-powered research assistant integrated with Google Gemini; accessed January 14, 2026.

[10] T. K. Chiu, "The impact of generative ai (genai) on practices, policies and research direction in education: A case of chatgpt and midjourney," *Interactive Learning Environments*, vol. 32, no. 10, pp. 6187–6203, 2024.

[11] C. Flaherty, "How ai is changing—not 'killing'—college," *Inside Higher Ed*, 2025.

[12] H. Bastani, O. Bastani, A. Sungu, H. Ge, Özge Kabakcı, and R. Mariman, "Generative ai without guardrails can harm learning: Evidence from high school mathematics," *Proceedings of the National Academy of Sciences*, vol. 122, no. 26, p. e2422633122, 2025. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.2422633122

[13] K. D. Wang, Z. Wu, L. T. II, C. Wieman, S. Salehi, and N. Haber, "Scaffold or crutch? examining college students' use and views of generative ai tools for stem education," 2024. [Online]. Available: https://arxiv.org/abs/2412.02653

[14] S. Krause, B. H. Panchal, and N. Ubhe, "Evolution of learning: Assessing the transformative impact of generative ai on higher education," *Frontiers of Digital Education*, vol. 2, no. 2, p. 21, 2025.

[15] P. Bassner, B. Lenk-Ostendorf, R. Beinstingel, T. Wasner, and S. Krusche, "Less stress, better scores, same learning: The dissociation of performance and learning in ai-supported programming education," *Computers and Education: Artificial Intelligence*, vol. 10, p. 100537, 2026. [Online]. Available: https://doi.org/10.1016/j.caeai.2025.100537

[16] M. Lehmann, P. B. Cornelius, and F. J. Sting, "Ai meets the classroom: When do large language models harm learning?" 2025. [Online]. Available: https://arxiv.org/abs/2409.09047

[17] W. Lyu, Y. Wang, T. R. Chung, Y. Sun, and Y. Zhang, "Evaluating the effectiveness of llms in introductory computer science education: A semester-long field study," in *Proceedings of the Eleventh ACM Conference on Learning @ Scale*, ser. L@S '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 63–74. [Online]. Available: https://doi.org/10.1145/3657604.3662036

[18] S. Melumad and J. H. Yun, "Experimental evidence of the effects of large language models versus web search on depth of learning," *PNAS Nexus*, vol. 4, no. 10, p. pgaf316, 10 2025. [Online]. Available: https://doi.org/10.1093/pnasnexus/pgaf316

[19] GitHub, "Github classroom," https://classroom.github.com/, 2025, assignment management and grading platform for education; accessed January 14, 2026.

[20] Respondus, Inc., "Lockdown browser," https://web.respondus.com/he/lockdownbrowser/, 2025, secure assessment browser for online exams; accessed January 14, 2026.