Supporting Active Learning in Computer Science with Live, Gamified Classroom Quizzes*

Andy Luu[1], Eli Young[1], and Michael Wehar[1,2]
[1]AlgoArena, [2]Haverford College
contact@algoarena.net

Abstract

Active learning measurably improves outcomes in STEM courses: across a meta-analysis of 225 studies, average examination scores rose by about 6% under active learning, and students in traditional lecture courses were 1.5 times more likely to fail. Tools such as Kahoot and Poll Everywhere are widely used to support active learning in classrooms, but they are subject-agnostic and lack question types tailored to computer science, such as code execution or diagram drawing. Platforms such as zyBooks, EdStem, and PrairieLearn support code execution and automated grading, but they are built for asynchronous, self-paced work rather than live, whole-class sessions. In this paper, we present AlgoArena Classroom, a live, gamified classroom platform that offers computer science question types, including coding challenges, code output prediction, time complexity analysis, and whiteboard diagramming, alongside standard multiple-choice and true/false formats. The system is built around a shared, real-time game state that keeps instructor and student views in sync, and supports both automated code execution for grading coding questions and AI-assisted grading for open-ended whiteboard questions. We describe the system's design, implementation, and the trade-offs made in building it, and we report on a first classroom pilot along with early independent adoption.

*Copyright ©2026 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

© CCSC, (2026). This is the author's version of the work. It is posted here by permission of CCSC for your personal use. Not for redistribution. The definitive version was published in The Journal of Computing Sciences in Colleges, 42, 5, http://dl.acm.org/.

1 Introduction

Active learning occurs when learners engage with the material (e.g., coding exercises or flashcards) [3, 5, 19]. Passive learning, on the other hand, means receiving information without actively interacting with it (e.g., watching a video). The effect of active learning is large and well established. A meta-analysis of 225 studies found that moving from traditional lecturing to active learning increased examination scores by roughly half a letter grade and cut failure rates by about a third [8, 9].

One way computer science (CS) classrooms try to encourage active learning is through gamification: using game design elements in non-game contexts to increase engagement [6, 12]. Instructors commonly use general-purpose tools like Kahoot and Poll Everywhere to run quick, competitive quizzes. However, these tools lack CS-specific features (like coding problems or stack diagram questions) that exercise the skills of writing and tracing code, which novice programmers often struggle with the most [1, 14, 15].

Two of us also felt this gap as students, when lecture-based courses did little to make the material exciting, and preparing for exams or interviews meant working through practice problems alone. The same problems, when attempted alongside classmates, with a timer running and a leaderboard on the projector, can become social, competitive, and genuinely fun.

To address this gap, we built AlgoArena Classroom for CS educators to host live, quiz-like sessions where students engage with course material. Classroom response systems like this have been studied as a way to increase student participation and provide instructors with immediate feedback during lectures [4]. Our research question is: can the live, competitive format of general-purpose quizzing tools and the CS question types of asynchronous platforms be combined into a single classroom tool, and what design trade-offs would that involve? We answer it by building and piloting such a system, making the following contributions:

- **A live, gamified classroom review system** with CS question types such as coding challenges executed against test cases, code output prediction, time complexity analysis, and whiteboard diagramming.
- **AI-assisted grading of hand-drawn diagrams**: each whiteboard drawing is graded by a large language model (LLM) that accepts images as soon as it is submitted, and the instructor can review and override any verdict, or opt out of AI grading entirely.
- **An architecture built around one shared game document**, keeping the instructor and student views in sync.

2 Related Work

2.1 Gamification Tools

Kahoot and Poll Everywhere are widely used tools for hosting live, whole-class quiz sessions [13, 22]. Students join with a code on their own devices, answer in real time, and see results after each question. These tools support multiple-choice and short-answer formats with game elements like points and leaderboards, which have been shown to increase student motivation and participation [12, 22].

2.2 CS-Oriented Tools

Several tools target CS instruction directly: zyBooks [11] and EdStem [7, 20] auto-grade programming assignments and code challenges that are completed in the browser; EdStem adds an asynchronous forum for course questions [18]; and PrairieLearn generates randomized variants of auto-graded coding and homework questions with adaptive scoring [24]. All three follow the same model, in which the platform automates much of the grading so students can work at their own pace with optional support from instructors and teaching assistants outside of class.

Neither approach meets all the needs of a CS classroom: general-purpose gamified tools bring the live, competitive format but provide little beyond multiple-choice and short-answer questions, whereas CS-oriented platforms have code execution and automated grading but only support asynchronous work. Our system combines the two.

3 System Design

3.1 Overview

We present a live classroom platform with a format similar to Kahoot, where an instructor prepares a set of questions ahead of time, hosts a session, and shares its join code (Figure 1). Once students have joined, the instructor starts the session for the whole class. Questions are drawn from a bank of roughly 70 curated classroom questions and more than 8,000 programming problems spanning about two dozen CS topics imported from AlgoArena's existing library of solo practice problems. Instructors can also create their own questions of the types below (manually or with an AI-assisted generator) and save their quizzes to share with others, allowing the platform to support course levels beyond data structures and algorithms (DSA).

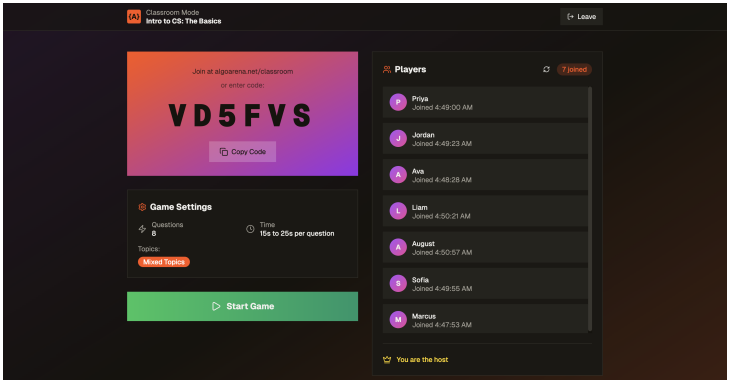

Figure 1: The host's lobby right before a live session begins: students have joined with the code shown, and the roster updates in real time until the host starts the session for the whole class.

3.2 Question Types

The system supports the standard multiple-choice and true/false formats along with several question types designed for CS:

- **Multiple Choice** and **True or False**: students select an answer from fixed options.
- **Code Output**: students predict the output of a given code snippet.
- **Time Complexity**: students identify the runtime complexity of an algorithm or code block.
- **Code Challenge**: students write and submit code, solving a short exercise or a full problem of the kind commonly used in technical interviews, drawn from AlgoArena's library, to be evaluated against a set of test cases (Figure 2).
- **Whiteboard**: students sketch a diagram (e.g., a tree or a stack diagram) in response to a prompt.

These formats deliberately span different levels of engagement in the Interactive, Constructive, Active, Passive (ICAP) framework, which orders learning activities from passively receiving material, through actively selecting among given options, to constructing something completely new [5]. Using this language, answering a Multiple Choice or True or False question is active engagement; predicting a program's output or complexity drives constructive code tracing; and Code Challenge and Whiteboard questions are fully constructive because students produce an artifact not given in the prompt. The platform also has other types of problems, including filling in missing code, ordering

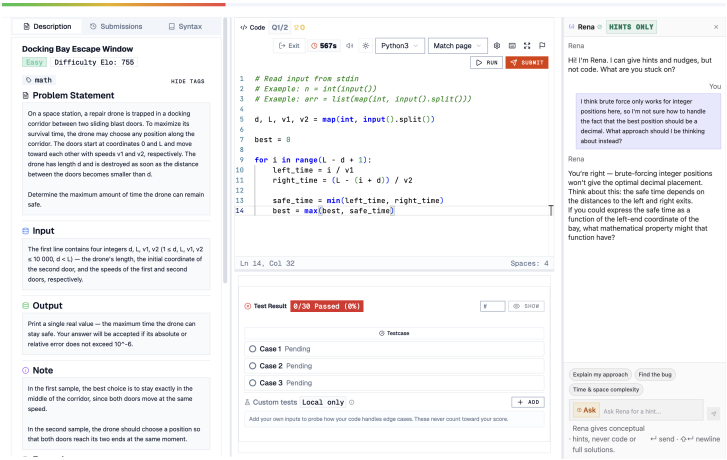

Figure 2: A coding question from the student's view: the problem statement (left), a code editor judged against test cases (center), and Rena (right), an optional AI assistant that gives hints without revealing the full solution.

the steps of an algorithm, identifying and fixing a bug, and Parsons problems, where students rearrange scrambled lines of code into a working program [17].

Each question can include an attached image, a configurable time limit, and a post-question explanation, which lets instructors use the platform for both assessment and teaching [2].

3.3 Session Flow

When an instructor hosts a session, all connected clients progress through a shared sequence of states: a lobby where students join, a countdown when the instructor starts the session, and then, for each round, a question, an answer reveal, and a leaderboard. Every transition is controlled by the host and instantly reflected on all student devices, keeping everyone in sync.

3.4 Real-Time Synchronization Architecture

The system's real-time behavior is built around a single shared game document rather than a dedicated WebSocket server (Figure 3). Each session is tracked by one document storing the game's current status, question index, question start time, connected players, and submitted answers. Clients subscribe to this document with real-time listeners, through which Firestore distributes each change live, so changes made by the host reach every connected device automatically.

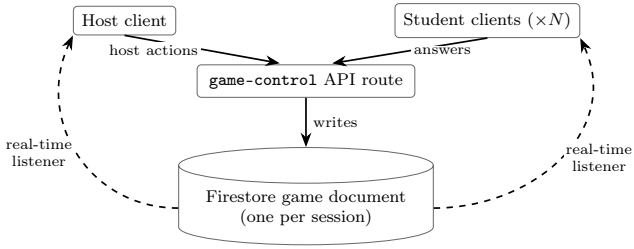

Figure 3: Real-time synchronization: host and student actions flow through the custom `game-control` API route into a single Firestore game document, from which every client renders using a real-time listener.

If a client's real-time connection drops, it falls back to polling the same game state every second, so the session stays synchronized on unreliable networks.

All in-game state changes (starting the countdown, advancing to the next question, submitting an answer) go through one control endpoint that updates the shared document, and every client, the host included, re-renders from it.

Question timers need to stay in sync across devices, so instead of running individual countdowns, which would start late on a slow connection, each client computes the time remaining using the question's recorded start timestamp and the current time, corrected by a one-time comparison with the server's clock.

3.5 Instructor View

Students see only the current question and their own submission status, while the instructor sees the full state of the session, which includes who has joined, a live count of submissions, and how the class answered once the question closes. This gives the instructor a live picture of the whole class's progress in the middle of the quiz, without looking at individual screens. After the session, the instructor can open a session report that displays results by question, with each student's score, accuracy, response time, and submitted code or drawings. Explanations shown after each question give students immediate feedback on their answers [2], while the session results show instructors what the class might need to review before the next lecture or exam, a benefit long documented for classroom response systems [4].

3.6 Student View

Students do not need an account or any installed software, since everything runs in the browser. Once the instructor starts the session, each question appears in its appropriate format, which includes a set of answer options, a code editor

for coding challenges, or a drawing canvas for whiteboard questions. For the six question types, students learn whether their answer was correct only at the answer reveal (excluding tests run during live coding questions), so a neighbor's screen shows at most a chosen answer, not whether it is right, while the question is still open. The reveal also shows the explanation (if present) along with the student's updated position on the leaderboard.

4 Implementation

4.1 Technology Stack

Our system is built with a Next.js and React frontend, written in TypeScript throughout, using a Monaco code editor (the one inside VS Code) for coding questions. The game state is stored in Firebase's Firestore, a cloud document database [10] (Section 3.4). A small set of Next.js API routes (`create-game`, `get-game`, `game-control`, `join-game`) handle writes to the document database, with `get-game` being used to serve clients that have fallen back to polling.

4.2 Code Execution and Grading

Automated assessment is common in CS education as a scalable way to give feedback on programming assignments [16], and our system judges Code Challenge submissions in a similar way. When a student submits code, our endpoint sends it to a hosted execution service (OneCompiler) that runs it against the problem's test cases. For most problems, this endpoint compares each test's output directly against the expected result, but problems with more than one correct answer (for instance, when there are multiple valid orderings) use a problem-specific checker function instead of a basic string comparison. Moreover, problems with especially large test suites are judged on the server, which loads the full test set from storage. All coding questions use the shared scoring formula outlined in Section 4.4, which gives partial credit for submissions that pass only some tests.

While solving a coding question, students can chat with Rena, an optional AI assistant (Figure 2) that offers conceptual hints designed not to reveal complete solutions or code.

4.3 Whiteboard Question Grading

Whiteboard questions, being open-ended and visual, are difficult to check deterministically, and manual grading of every drawing in the middle of the quiz

would be too slow, so grading for these questions is AI-assisted. When a student's drawing is completed, it is converted to a PNG, submitted, and graded immediately by an automated grading step that sends the drawing (together with any reference image) to an LLM that accepts images (Claude Sonnet 5 via Amazon Bedrock at the time of writing) and generates a pass/fail verdict and a one-sentence rationale, which are stored on the server until revealed. Other automated diagram assessment systems grade by matching the components of a student's diagram, one by one, against a set of correct solutions [21], whereas our system asks AI models to judge each diagram as a whole. The instructor can review model verdicts and rationales as they arrive, and can override any outcomes before advancing the quiz to the leaderboard. Points are awarded once a drawing is marked correct, using the same scoring formula as other question types (Section 4.4). In the platform's solo practice mode, where an instructor is not present, LLM verdicts are treated as final.

4.4 Overall Scoring

Scoring is determined by a base score that can be raised by bonuses earned through speed and consecutive correct answers (Table 1). The speed bonus scales linearly with the fraction of the time limit that is remaining, where an immediate, fully correct answer earns up to 1,000 points. The streak bonus adds 100 points per consecutive correct answer beyond the first, with a cap of 300 points. Code Challenge submissions are special in that they earn base points in proportion to the number of test cases they pass, but a submission that passes only some test cases earns no speed or streak bonus, with the student's streak resetting to 0. Every other question type uses the default formula, which combines base points, speed bonuses, and streak bonuses.

Notably, awarding points for speed is documented as a source of stress in the literature on Kahoot and similar systems [22], so three purposeful design choices soften it here: (1) per-question time limits let instructors de-emphasize speed when a careful answer matters, (2) the streak cap limits how far ahead a single streak can carry one student, and (3) a correct answer earns its base score regardless of speed. For privacy, students can join anonymously under a chosen nickname, and drawings are sent for grading without personally identifying information attached.

5 Discussion

5.1 Design Trade-Offs

We chose Firestore's real-time listeners instead of running a dedicated WebSocket server. While a WebSocket server would give us more control over when

Table 1: Scoring varies only slightly by the question type. The speed bonus is $500 \times (1 - \mathit{elapsed}/\mathit{limit})$, where *limit* is the question's time limit. The streak bonus is $100 \times \min(\mathit{streak} - 1, 3)$.

Question type	Base	Speed bonus	Streak bonus
Option-based[a]	500	up to 500	up to 300
Code Challenge (all tests pass)	500	up to 500	up to 300
Code Challenge (p of t tests pass)	$500 \times p/t$	—	—
Whiteboard (marked correct)	500	up to 500	up to 300

[a]Multiple Choice, True or False, Code Output, and Time Complexity.

updates reach clients, we would have to write and maintain the code that tracks who is connected, reconnects any stragglers, and delivers each change. Firestore handles all of that for us, with one constraint: it provides updates to clients as soon as possible, without batching, delays, or prioritization. This keeps the system simple and reliable for any given class.

Routing in-game state changes through one game document and one control endpoint has a similar trade-off: while separate update paths (for example, the host updating its own view and pushing changes to students) would be more efficient, a subtle bug in either path could leave the host and students out of sync. Our single update path, however, prevents that, at the cost of re-rendering all clients whenever any part of the document changes, whether it shows that part or not. This cost is negligible even for most large lectures.

Similarly, AI-assisted grading of whiteboard questions trades accuracy for speed, as AI models cannot consistently replicate an instructor's judgment, particularly for unconventional or partially correct diagrams.

5.2 Preliminary Classroom Use

The platform was piloted in an undergraduate course at Bryn Mawr College, Competitive Coding Algorithms (CS283), taught by one of the authors. A live, in-class practice quiz was held on April 23, 2026. All seven undergraduates present completed a six-question quiz in 30 minutes: four short questions (one multiple-choice and three true/false) with limits of 10 to 20 seconds, two of them light-hearted polls, and two full coding questions. The full coding questions were limited to 20 minutes and were judged against 77 and 14 test cases, respectively. The course used AlgoArena's practice library throughout the semester, alongside LeetCode, NeetCode, Project Euler, and Kattis [23]. This pilot was the course's only live practice quiz of this format. Class-wide accuracy on the four short questions on course material averaged 64%, and each coding question was fully solved by three students, all in Java, in un-

der 11 minutes. In an informal, anonymous survey after the session, shared with the students' consent, five of seven respondents said that the format has potential for CS classes and is fun and useful for reviewing lecture concepts. Criticisms included unclear problem statements, not being able to see every test case, and, from one student, a preference for practicing alone. Within days, in response, we enabled custom test cases for Code Challenge questions, let instructors preview prospective problems, and added a light mode. Beyond this pilot, twelve hosts unaffiliated with the project ran sixteen live quizzes from February to September 2026, including three in Spanish, showing that instructors can create quizzes in the language of instruction rather than being limited to English.

5.3 Limitations

The pilot was too small to make general claims about any resulting improvement in engagement or learning. No course has used the live quizzes consistently throughout a term, and we did not measure student learning before and after the pilot. Additionally, the trade-offs from Section 5.1 have not been evaluated empirically. AI-assisted grading of whiteboard questions has not been validated against human graders, but we suspect its effectiveness decreases as diagrams get more complex. We have also not measured how a pause for grading in a fast-paced quiz affects student concentration, engagement, or performance. Finally, the built-in question bank mainly covers DSA, so instructors of other courses would currently have to create most of their questions.

6 Conclusion

We have presented AlgoArena Classroom, a working, piloted system that combines live, gamified sessions with CS question types. This system has been built around a shared, real-time game state, automated code execution, and AI-assisted diagram grading. A full-course evaluation measuring engagement and learning outcomes over time, along with a comparison of AI-assisted diagram grading against human graders, is our natural next step. As additional future work, we would like to add question banks for upper-level courses (e.g., Compilers or Theory of Computation) and evaluate more in-depth class management features. Instructors are encouraged to try the platform and to contact us with any feedback.

Acknowledgments. We thank the pilot students and the anonymous reviewers. Parts of this manuscript were drafted and edited using AI tools (e.g., Anthropic's Claude) with careful review and verification by the authors.

References

[1] João Henrique Berssanette and Antonio Carlos de Francisco. "Active Learning in the Context of the Teaching/Learning of Computer Programming: A Systematic Review". In: *Journal of Information Technology Education: Research* 20 (2021), pp. 201–220. DOI: 10.28945/4767.

[2] Paul Black and Dylan Wiliam. "Assessment and Classroom Learning". In: *Assessment in Education: Principles, Policy & Practice* 5.1 (1998), pp. 7–74. DOI: 10.1080/0969595980050102.

[3] Charles C. Bonwell and James A. Eison. *Active Learning: Creating Excitement in the Classroom*. ASHE-ERIC Higher Education Report No. 1. Washington, DC: ERIC Clearinghouse on Higher Education, 1991. 121 pp. ISBN: 1-878380-08-7.

[4] Jane E. Caldwell. "Clickers in the Large Classroom: Current Research and Best-Practice Tips". In: *CBE—Life Sciences Education* 6.1 (2007), pp. 9–20. DOI: 10.1187/cbe.06-12-0205.

[5] Michelene T. H. Chi and Ruth Wylie. "The ICAP Framework: Linking Cognitive Engagement to Active Learning Outcomes". In: *Educational Psychologist* 49.4 (2014), pp. 219–243. DOI: 10.1080/00461520.2014.965823.

[6] Sebastian Deterding et al. "Gamification: Using Game-Design Elements in Non-Gaming Contexts". In: *CHI '11 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '11. Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 2425–2428. ISBN: 9781450302685. DOI: 10.1145/1979742.1979575.

[7] Ed. *Ed Lessons*. https://edstem.org/lessons. Accessed July 12, 2026.

[8] Scott Freeman et al. "Active learning increases student performance in science, engineering, and mathematics". In: *Proceedings of the National Academy of Sciences* 111.23 (2014), pp. 8410–8415. DOI: 10.1073/pnas.1319030111.

[9] Karla J. Gingerich et al. "Active Processing via Write-to-Learn Assignments: Learning and Retention Benefits in Introductory Psychology". In: *Teaching of Psychology* 41.4 (2014), pp. 303–308. DOI: 10.1177/0098628314549701.

[10] Google. *Get Real-time Updates with Cloud Firestore*. https://firebase.google.com/docs/firestore/query-data/listen. Accessed July 11, 2026.

[11] Chelsea Gordon, Roman Lysecky, and Frank Vahid. "The Rise of Program Auto-grading in Introductory CS Courses: A Case Study of zyLabs". In: *2021 ASEE Virtual Annual Conference Content Access Proceedings*. 2021. URL: https://peer.asee.org/37887.

[12] Juho Hamari, Jonna Koivisto, and Harri Sarsa. "Does Gamification Work? – A Literature Review of Empirical Studies on Gamification". In: *2014 47th Hawaii International Conference on System Sciences*. 2014, pp. 3025–3034. DOI: 10.1109/HICSS.2014.377.

[13] Wendi M. Kappers and Stephanie L. Cutler. "Poll Everywhere! Even in the Classroom: An Investigation into the Impact of Using PollEverywhere in a Large-Lecture Classroom". In: *Computers in Education Journal* 6.20 (2015), pp. 140–145. URL: https://commons.erau.edu/publication/333.

[14] Raymond Lister et al. "A multi-national study of reading and tracing skills in novice programmers". In: *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*. ITiCSE-WGR '04. Leeds, United Kingdom: Association for Computing Machinery, 2004, pp. 119–150. ISBN: 9781450377942. DOI: 10.1145/1044550.1041673.

[15] Kasia Muldner, Jay Jennings, and Veronica Chiarelli. "A Review of Worked Examples in Programming Activities". In: *ACM Transactions on Computing Education* 23.1 (2023). DOI: 10.1145/3560266.

[16] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. "Automated Assessment in Computer Science Education: A State-of-the-Art Review". In: *ACM Transactions on Computing Education* 22.3 (2022). DOI: 10.1145/3513140.

[17] Dale Parsons and Patricia Haden. "Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses". In: *Proceedings of the 8th Australasian Computing Education Conference (ACE '06)*. Australian Computer Society, 2006, pp. 157–163.

[18] Christopher Piech et al. "Code in Place: Online Section Leading for Scalable Human-Centered Learning". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. SIGCSE '21. Virtual Event, USA: Association for Computing Machinery, 2021, pp. 973–979. ISBN: 9781450380621. DOI: 10.1145/3408877.3432562.

[19] Michael Prince. "Does Active Learning Work? A Review of the Research". In: *Journal of Engineering Education* 93.3 (2004), pp. 223–231. DOI: 10.1002/j.2168-9830.2004.tb00809.x.

[20] Chakkrit Tantithamthavorn and Norman Chen. "Unit Testing Challenges with Automated Marking". In: *2023 30th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2023, pp. 544–548. DOI: 10.1109/APSEC60848.2023.00067.

[21] Pete G. Thomas, Neil Smith, and Kevin G. Waugh. "Computer assisted assessment of diagrams". In: *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*. June 2007, pp. 68–72. DOI: 10.1145/1268784.1268806. URL: https://oro.open.ac.uk/19373/.

[22] Alf Inge Wang and Rabail Tahir. "The effect of using Kahoot! for learning – A literature review". In: *Computers & Education* 149 (2020), p. 103818. ISSN: 0360-1315. DOI: 10.1016/j.compedu.2020.103818.

[23] Michael Wehar. *The State of Coding Challenge Problems (2026)*. https://michaelwehar.wordpress.com/2026/05/24/the-state-of-coding-challenge-problems-2026/. Blog post, May 24, 2026. Accessed September 2, 2026.

[24] Matthew West, Geoffrey Herman, and Craig Zilles. "PrairieLearn: Mastery-Based Online Problem Solving with Adaptive Scoring and Recommendations Driven by Machine Learning". In: *2015 ASEE Annual Conference and Exposition Proceedings*. ASEE Conferences, 2015. DOI: 10.18260/p.24575.